マルチグリッド法に基づく
高性能ポアソンソルバに関する研究

河合　直聡

目 次

第1章　序論

1.1　はじめに

近年，科学技術分野における計算機を使ったシミュレーションの重要性が高まっている．計算機シミュレーションは，従来から測定や実験が困難な物理現象の観測・予測や，工業製品の設計・検証コストの削減などの用途で用いられてきたが，計算機関連技術の進展に伴いより大規模・複雑かつ高精度な解析が可能となり，その応用分野が広がっている．たとえば，複数の物理現象を同時に扱うマルチフィジックスシミュレーションが最近注目されており，流体と電磁場を同時に扱うプラズマシミュレーション [1] や，電気生理学的現象と構造解析を同時に扱う心臓シミュレーション [2] など，さまざまな分野で研究開発が盛んに行われている．本稿では，これらのシミュレーションでしばしば計算の中核をなす，ポアソン方程式の高速な求解法（ソルバ）の研究成果について述べる．

ポアソン方程式は楕円型の二階偏微分方程式であり，さまざまな物理現象の基礎方程式として用いられている．たとえば，電磁気学では電荷分布から静電ポテンシャルを導く方程式として，また流体力学では流速から圧力分布を求める方程式として用いられている．したがって，ポアソン方程式の求解を内部に持つシミュレーションは数多く存在し，離散化による数値解法の研究も盛んに行われている．離散化手法としては有限差分法，有限要素法，境界要素法などがあるが，空間を三角形要素や四面体要素などで分割した非構造格子を使って離散化する有限要素法は，複雑な形状を持つ対象をモデル化した場合でも少ない格子数で高い精度の結果を得ることができるため，特にパーソナルコンピュータなどの小規模な計算環境に適した解法として広く利用されている．

一方でマルチフィジックスシミュレーションでは，複数の物理現象の相互作用を自然に表現するために，現象間の共通インタフェースという観点で規則的な構造格子を採用し，有限差分法による離散化を行うことが多い．しかし，構造格子に基づく離散化を複雑な構成要素を持つシミュレーションに適用して高い精度を得るためには，格子間隔を小さくした大規模な格子を用いる必要があり，結果的に膨大な次元数の連立一次方程式を解くことが求められる．一般に連立一次方程式の求解には大きな計算時間を要し，かつ問題規模の増大が計算時間の増加に直結することから，マルチフィジックスシミュ

レーション全体の計算時間のほとんどをポアソン方程式の求解に費やすこともしばしばある．

このように大きな計算量を必要とする問題に対しては，並列計算によって計算時間の短縮と問題規模の拡大を実現することが一般的になっており，規則的な構造格子は問題の分割が比較的容易であることから，並列計算に適しているということもできる．またたとえば，世界のスーパーコンピュータの中で 500 位までの性能を持つものを定期的にリストするプロジェクトである TOP500[3] によると，2013 年 11 月時点の上位 30 位までのスーパーコンピュータが保有するコア数はいずれも 10 万を超えており，システムを構成する個々のプロセッサあたりのコア数も 6 以上となっている．さらに Intel 社が 2012 年に発表したプロセッサ Xeon Phi[4] は，60 個のコアを有している上にコアあたり 4 スレッドの実行が可能であることから，プロセッサあたり 240 ものスレッドによる並列計算を行うことができ，これを採用した世界最大・最速のスーパーコンピュータである Tianhe2 は，300 万を超えるコアと 1000 万を超えるスレッドという膨大なスケールの並列計算環境となっている．このような大規模並列環境を利用して連立一次方程式を高速に解くためには，解法の改善などによる逐次性能の向上もさることながら，効率的な並列化が必須要件である．特に，分散メモリ環境を考慮した並列化は問題の大規模化に寄与することもでき，シミュレーションの高精度化という観点からも望ましい．

そこで本稿では，規則的な構造格子に基づいてポアソン方程式を離散化して得られる連立一次方程式を，さまざまな並列計算環境で高速に解くための方法として，幾何マルチグリッド法の並列化について議論する．離散化されたポアソン方程式の求解法には，一般的な連立一次方程式の求解法である直接法と，SOR 法 [5] や CG 法 [6] などの反復法があり，また FFT(高速フーリエ変換) もしばしば用いられる [7]．これらの中でガウス消去法などの直接法は，未知変数の数 N に対して $\mathcal{O}(N^2)$ の空間計算量と $\mathcal{O}(N^3)$ の時間計算量を要するため，たとえば本稿で述べる数値実験の中で最小規模の問題でも N が 1 億を超えることから，ごく小規模な問題を除いて非現実的な解法である．SOR 法や CG 法は，ポアソン方程式の求解での空間計算量がいずれも $\mathcal{O}(N)$ であるため現実的であるが，1 回あたりの時間計算量が $\mathcal{O}(N)$ の反復操作の回数が N の増加にしたがって増える傾向にあり，並列計算を行っても問題規模の拡大が求解時間の増加に繋がるという欠点がある．FFT は $\mathcal{O}(N)$ の空間計算量と $\mathcal{O}(N \log N)$ の時間計算量という優れた性質を持つが，連立一次方程式の係数に対する制約が強く，適用可能な物理現象の範囲が限定されるという問題がある．また並列計算の観点では，多数のプロセスに分散配置された巨大な 3 次元配列に対する転置操作を行うため，全プロセスあるいは複数のプロセス群の中での全対全通信に要する時間が，特に大規模並列環境では重大なボトルネックとなることが知られている [8].

これらに対して，マルチグリッド法は空間計算量が $\mathcal{O}(N)$ であるだけでなく，理想的な場合には 1 回あたりの時間計算量が $\mathcal{O}(N)$ であり，かつ反復操作の回数が N に依存しないこと，すなわち同種の問題の格子間隔を狭める，あるいはより大きな空間を対象とすることで大規模化しても，時間計算量が $\mathcal{O}(N)$ であるという極めて優れた特徴を持っている．このことは，適切な並列化を行うことができれば，問題規模の拡大に応じて計算環境の並列度を増やすことで，問題規模によらず一定の時間で求解できるということも意味しており，高性能システムの並列度の急速な向上にも適合している．

本稿では，このマルチグリッド法の並列化，特に一連の処理手順の中でも収束性と並列計算性能に強く影響するスムージング処理の並列化について議論する．また本稿での議論の中心的な課題は，優れた収束性と並列計算性能を併せ持つ新たなスムーザの提案と，このスムーザを用いたマルチグリッド法ソルバが，共有メモリ並列環境，分散メモリ並列環境，および Xeon Phi のそれぞれにおいて優れた性能を発揮することの実証・評価である．

1.2 本稿の構成

まず第 2 章では，本論である第 3 章以降の議論の準備として，本稿の主題である幾何マルチグリッド法によるポアソン方程式による求解について議論する．まずポアソン方程式とその有限差分法による離散化により得られる連立一次方程式を定式化した後，マルチグリッド法の重要な構成要素であるスムージングに用いられる定常反復法について述べる．具体的には，代表的な定常反復法であるヤコビ法（Jc 法）とその変形である重み付きヤコビ法（w-Jc 法），およびガウスザイデル法（GS 法）を取り上げ，それらの収束性についても解析的に示す．続いて幾何マルチグリッド法の処理手順を詳細に示し，その収束性とスムーザの関係についても議論する．さらに本稿で示すマルチグリッド法の実装で共通して用いるデータ構造を定義した後，連立一次方程式の反復解法，マルチグリッド法，およびスムーザの並列化手法に関する既存の研究について議論する．

続く第 3 章から第 5 章が本稿の中核であり，さまざまな並列計算環境に適合した並列スムーザの提案とその優秀性の実証評価を行う．まず第 3 章では，マルチグリッド法の並列化での重要なポイントであるスムーザの並列化手法として，GS 法にブロック化赤-黒順序付け法を適用したブロック化赤-黒順序付けガウスザイデル (BRB-GS) スムーザと，それをさらに改良して参照局所性と収束性を向上させた改良型ブロック化赤-黒順序付けガウスザイデル (mBRB-GS) スムーザを提案する．マルチグリッド法の並列化スムーザの既存手法としては，赤-黒順序付けガウスザイデル (RB-GS) スムーザと，w-Jc 法と GS 法を併用したハイブリッド (Hybrid) スムーザがあるが，前者は配列に対するストライドアクセスによりスムージング一回あたりの計算時間が長

くなり，後者は w-Jc 法の併用による収束性の悪化が問題となる．これに対して BRB-GS スムーザは，解析格子を分割して得たブロックに対して赤-黒順序付け法を適用する手法であるため，収束性は逐次 GS スムーザあるいは RB-GS スムーザと同程度であると期待でき，各ブロック内のスムージングには逐次 GS 法を用いることから RB-GS スムーザの問題点であるストライドアクセスも回避することができる．また mBRB-GS スムーザは，ブロック単位の逐次 GS スムージングを複数回反復することでマルチグリッド法ソルバ全体の収束性を向上させつつ，キャッシュ容量を考慮したブロックサイズの設定により 2 回目以降の反復に要する時間を 1 回目の反復よりも大幅に短縮することで，マルチグリッド法全体の計算時間を短縮するものである．この BRB-GS と mBRB-GS スムーザを用いたマルチグリッド法ソルバのマルチコアプロセッサでの性能を評価し，既存のスムーザによるものに比べて大きな優位性を持つことを示す．

第 4 章では，上記の BRB-GS および mBRB-GS スムーザを用いたマルチグリッド法ソルバの，分散メモリ並列環境での性能について議論する．第 3 章で示した参照局所性と収束性の観点に加え，分散メモリ並列環境に対応したプロセス並列化ではプロセス間通信の回数やデータ量も性能に大きな影響を与える．この章では，二つの提案手法と二つの既存手法との間で通信性能を左右するような差はないことを並列プログラムの構造から導き，この結果をプロセス並列環境の性能でも提案手法が既存手法を大幅に上回ることを示して実証する．

第 5 章では，メニーコアプロセッサ Xeon Phi を対象とした mBRB-GS スムーザの最適化について議論する．Xeon Phi を用いて高い性能を得るためには，その特徴である 512 ビット幅の SIMD 演算を有効に活用しなければならない．しかし mBRB-GS スムーザでのブロック単位の逐次 GS スムージングにはループ運搬フロー依存性があり，自然な実装を行うと SIMD 演算が全く利用できない．そこで GS スムージングの最内ループ内の計算の中で SIMD 並列化を阻害しているのが一つの加算項のみであることに着目し，加算項ごとのループ分割をキャッシュの有効利用を意識しつつ実施することで，分割後に得られる 6 つのループの中の 5 つに対して SIMD 並列化が適用できることを示す．またこの手法が mBRB-GS スムーザを用いたマルチグリッド法ソルバの性能を向上させることと，SIMD 並列化が容易な既存スムーザによるソルバよりも高い性能が得られることも示す．

最後に第 6 章では本稿に示す研究成果を総括するとともに，この研究の深化・発展や研究過程で見出された新たな方向性について議論する．

第2章　マルチグリッド法によるポアソン方程式の差分解析

本稿では，3 次元ポアソン方程式を構造格子空間と有限差分法を用いて離散化して得られる連立一次方程式を，高速に解くためのライブラリであるポアソンソルバについて論じる．その解法としてマルチグリッド法を採用しているが，これはマルチグリッド法が理想的な条件下では対象問題の規模に収束性が依存しないという特徴をもっているためである．本章ではポアソン方程式の離散化と定常反復法について述べた後，マルチグリッド法とその収束性，本稿で議論するソルバでのデータ格納形式，および関連研究について述べる．

2.1　3 次元ポアソン方程式の差分解析

本節では有限差分法を用いたポアソン方程式の離散化について述べる [9].

2.1.1　ポアソン方程式

ポアソン方程式は楕円型の二階偏微分方程式であり，ポアソン方程式が適用される空間 Ω の中で拡散係数が一定である場合，一般に以下の式で与えられる．

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}+\frac{\partial^2\phi}{\partial z^2}=-\rho \quad \text{on } \Omega \tag{2.1}$$

式 (2.1) の ρ は既知関数を，ϕ は未知関数を表す．ここで本稿で議論する空間 Ω は，x，y，z 各方向の長さがそれぞれ $LenX$，$LenY$，$LenZ$ の直方体形状であるとする．またその空間の境界部分 $\partial\Omega$ 上の ϕ に対して，以下の固定境界条件を付加する．

$$\phi = 0 \text{ on } \partial\Omega \tag{2.2}$$

2.1.2　有限差分法によるポアソン方程式の離散化

本節では式 (2.1) を有限差分法を用いて離散化し，連立一次方程式を得るまでを扱う．

有限差分法を用いて離散化を行うためには，既知と未知関数を離散空間に配置する必要がある．一般に 3 次元空間の離散化は，対象とする領域を多面体の集合で覆い，その頂点，辺，面あるいは多面体自体に空間的に離散化された変数を配置することにより行われる．本稿では，空間 Ω を単位直方体（あるいは立方体）で覆うことで格子状に分割した，構造格子を用いた離散化を行う．また各直方体の頂点，すなわち 3 方向の格子が交わる格子点に，整数座標 (i,j,k) $(1 \leq i \leq NX,\ 1 \leq j \leq NY, 1 \leq k \leq NZ)$ を与える．この整数座標空間 $[1,NX] \times [1,NY] \times [1,NZ]$ を $\Omega^{\{h\}}$ と呼び，格子点における式 (2.1) の関数 ϕ および ρ の値をそれぞれ $\phi_{i,j,k}$ および $\rho_{i,j,k}$ と表記する．また，$i \in \{1,NX\}$，$j \in \{1,NY\}$，$k \in \{1,NZ\}$ のいずれかを満たす格子点座標，すなわち $\partial\Omega$ に含まれる座標のものを除いた $\phi_{i,j,k}$ および $\rho_{i,j,k}$ で構成されたベクトルを，それぞれ $\boldsymbol{\phi}$ および $\boldsymbol{\rho}$ と表記する．よってこれらのベクトルの要素数は $(NX-2) \times (NY-2) \times (NZ-2)$ となる．なお，格子点 $(i,j,k) \in [2,NX-1] \times [2,NY-1] \times [2,NZ-1]$ と $\boldsymbol{\phi}$ や $\boldsymbol{\rho}$ の要素を対応付けるために，辞書式順序付けを定める以下の関数 $lex(i,j,k)$ を導入する．

$$lex(i,j,k) = \begin{cases} (i-1)+(NX-2)(j-2)+(NY-2)(NX-2)(k-2) \\ \qquad\qquad (i,j,k) \in [2,NX-1] \times [2,NY-1] \times [2,NZ-1] \\ 0 \qquad\qquad \text{otherwise} \end{cases} \tag{2.3}$$

この関数により $\boldsymbol{\phi}$ や $\boldsymbol{\rho}$ の l 番目の要素と，$l = lex(i,j,k)$ なる格子点 (i,j,k) とが一対一に対応付けられる．図 2.1 に，$NX = NY = NZ = 5$ の構造格子の例を示す．

次に $\boldsymbol{\phi}$，$\boldsymbol{\rho}$ を用いて二階偏微分を表現する．そのためにまず，未知関数 ϕ は無限回微分可能な関数であるとし，テーラー展開を行う．一般に $x = a$ でのテーラー展開は以下の式で与えられる．

$$\phi(x,y,z) = \phi(a,y,z) + \sum_{m=1}^{\infty} \frac{1}{m!} \frac{\partial^m \phi}{\partial x^m}(a,y,z) \cdot (x-a)^m \tag{2.4}$$

この式に基づき格子点 $(i \pm 1,j,k)$ が格子点 (i,j,k) に十分近接しているとして，

$$\Delta x = \frac{LenX}{NX-1}$$

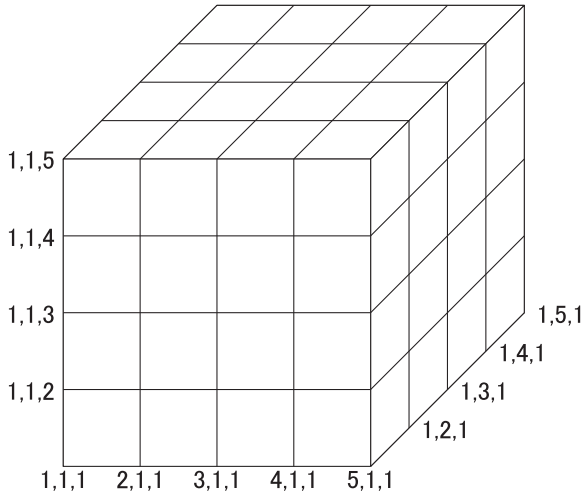


図 2.1: $NX = NY = NZ = 5$ の場合の構造格子

を用い，以下の式を得る．

$$\phi_{i\pm1,j,k} = \phi_{i,j,k} + \sum_{m=1}^{\infty} \frac{1}{m!} \frac{\partial^m \phi_{i,j,k}}{\partial x^m} (\pm\Delta x)^m \tag{2.5}$$

$$\phi_{i+1,j,k} = \phi_{i,j,k} + \frac{\partial \phi_{i,j,k}}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 \phi_{i,j,k}}{\partial x^2}(\Delta x)^2 + \frac{1}{6}\frac{\partial^3 \phi_{i,j,k}}{\partial x^3}(\Delta x)^3 + \frac{1}{24}\frac{\partial^4 \phi_{i,j,k}}{\partial x^4}(\Delta x)^4 \cdots \tag{2.6}$$

$$\phi_{i-1,j,k} = \phi_{i,j,k} - \frac{\partial \phi_{i,j,k}}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 \phi_{i,j,k}}{\partial x^2}(\Delta x)^2 - \frac{1}{6}\frac{\partial^3 \phi_{i,j,k}}{\partial x^3}(\Delta x)^3 + \frac{1}{24}\frac{\partial^4 \phi_{i,j,k}}{\partial x^4}(\Delta x)^4 \cdots \tag{2.7}$$

最後に，式 (2.6) と式 (2.7) の和を取り，変形すると以下の式を得る．

$$\phi_{i+1,j,k} + \phi_{i-1,j,k} = 2\phi_{i,j,k} + \frac{\partial^2 \phi_{i,j,k}}{\partial x^2}(\Delta x)^2 + \frac{1}{12}\frac{\partial^4 \phi_{i,j,k}}{\partial x^4}(\Delta x)^4 \cdots \tag{2.8}$$

$$\begin{aligned} \frac{\partial^2 \phi_{i,j,k}}{\partial x^2} &= \frac{\phi_{i+1,j,k} + \phi_{i-1,j,k} - 2\phi_{i,j,k}}{(\Delta x)^2} - \frac{1}{12}\frac{\partial^4 \phi_{i,j,k}}{\partial x^4}(\Delta x)^2 \cdots \\ &\approx \frac{\phi_{i+1,j,k} + \phi_{i-1,j,k} - 2\phi_{i,j,k}}{(\Delta x)^2} \end{aligned} \tag{2.9}$$

式 (2.9) は，NX を大きな値として Δx を十分微小な値とすることにより，$\phi_{i,j,k}$ の 2 階微分を第 1 項のみで表現する近似，すなわち第 2 項以降を無視した近似が，十分な精度で行えることを意味している．

次に，有限差分法によって離散化された 3 次元ポアソン方程式を求める．式 (2.1) の y および z 方向も x 方向と同様に離散化すると，式 (2.1) は

$\Delta y = LenY/(NY-1)$，$\Delta z = LenZ/(NZ-1)$ として以下のように表される．

$$\frac{\phi_{i+1,j,k}+\phi_{i-1,j,k}-2\phi_{i,j,k}}{(\Delta x)^2}+\frac{\phi_{i,j+1,k}+\phi_{i,j-1,k}-2\phi_{i,j,k}}{(\Delta y)^2} + \frac{\phi_{i,j,k+1}+\phi_{i,j,k-1}-2\phi_{i,j,k}}{(\Delta z)^2}=-\rho_{i,j,k} \tag{2.10}$$

式 (2.1) のような偏微分方程式を式 (2.10) のような差分方程式で近似する方法は，有限差分法と呼ばれる．特に，式 (2.10) は $\phi_{i,j,k}$ を中心としてそれと隣接する 6 点の格子点の値を用いて記述されているため，7 点差分法と呼ばれる．なおより多くの格子点を用いて差分方程式を構成する方法もあるが，本稿では最も広く使われている 7 点差分法を対象とする．

式 (2.10) は全ての格子点で成り立つため，全ての格子点に関する方程式が連立している．よって，$\boldsymbol{\phi}$ の値を $\boldsymbol{\rho}$ から得るためには，$(NX-2)\times(NY-2)\times(NZ-2)$ の未知変数を持った連立方程式を解く必要がある．この連立一次方程式を以下のように記述する．

$$\boldsymbol{A\phi}=\boldsymbol{\rho} \tag{2.11}$$

ここで係数行列 $\boldsymbol{A}$ は，4 種類の値の非零要素が各行に高々7 個存在する疎行列となる．具体的には非零要素の値をそれぞれ a，b，c，d とし，$l=lex(i,j,k)$ とすると，A の要素 $A_{l,m}$ は以下のように表される[1]．

$$A_{l,m}=\begin{cases} a=-1/(\Delta z)^2 & m=lex(i,j,k\pm1)\\ b=-1/(\Delta y)^2 & m=lex(i,j\pm1,k)\\ c=-1/(\Delta x)^2 & m=lex(i\pm1,j,k)\\ d=2\left\{1/(\Delta x)^2+1/(\Delta y)^2+1/(\Delta z)^2\right\} & m=l\\ 0 & \text{otherwise}\end{cases} \tag{2.12}$$

本稿で議論する高速な求解法は，この連立一次方程式を対象とする．

2.2 定常反復法

本節では，2.1 節で導出した連立一次方程式を解く手法の一つである定常反復法について述べる．定常反復法は，ある同一の反復操作をある初期値に対して複数回行うことで，近似解を得る手法である．定常反復法の代表的なものとしてヤコビ (Jc) 法やガウスザイデル (GS) 法がある．本稿が対象とするマルチグリッド法はこれらの手法を応用した手法であり，その処理手順の中でもこれらの定常反復法を用いるため，先に定常反復法について述べる [10].

[1]境界 $\partial\Omega$ に隣接する格子点，たとえば $(2,2,2)$ については，$(1,2,2)$ などの境界上の格子点に対応する列が存在しない ($lex(1,2,2)=0$) ため，これらを除く内部格子点に対応する列の要素のみが非零となる．したがって，たとえば行 $l=lex(2,2,2)=1$ の非零要素は 4 個である．

2.2.1 ヤコビ法とガウスザイデル法

まず，ヤコビ法（Jc 法）の反復操作について述べる．連立一次方程式の係数行列 $\boldsymbol{A}$ を，以下のように対角行列 $\boldsymbol{D}$, 狭義上三角行列 $\boldsymbol{U}$ および狭義下三角行列 $\boldsymbol{L}$ に分解する．

$$\boldsymbol{A} = (\boldsymbol{D} - \boldsymbol{U} - \boldsymbol{L}) \tag{2.13}$$

一般的な Jc 法の更新手順は，何らかの方法で定めた初期解 $\boldsymbol{\phi}^{(0)}$ に対して反復手順を m 回適用した後の近似解を $\boldsymbol{\phi}^{(m)}$ として，以下のように表される．

$$\boldsymbol{\phi}^{(m+1)} = \boldsymbol{D}^{-1}\left\{\boldsymbol{\rho} + (\boldsymbol{U} + \boldsymbol{L})\,\boldsymbol{\phi}^{(m)}\right\} \tag{2.14}$$

特に，本稿で対象とする連立一次方程式は式 (2.12) で示した係数行列を持つため，以下のように表すことができる．

$$\begin{aligned}\phi_{i,j,k}^{(m+1)} = \frac{1}{d}\Big(&\rho_{i,j,k} + a\cdot\phi_{i,j,k-1}^{(m)} + b\cdot\phi_{i,j-1,k}^{(m)} + c\cdot\phi_{i-1,j,k}^{(m)} \\ &+ c\cdot\phi_{i+1,j,k}^{(m)} + b\cdot\phi_{i,j+1,k}^{(m)} + a\cdot\phi_{i,j,k+1}^{(m)}\Big)\end{aligned} \tag{2.15}$$

Jc 法がある条件下で収束すること，すなわちこの手順の適用回数 m を大きくすればするほど $\boldsymbol{\phi}^{(m)}$ が厳密解に近くなることが知られている．特に本稿で扱う 3 次元ポアソン方程式の固定境界値問題を差分近似した連立一次方程式では，Jc 法の収束性が保証されている．

次に，ガウスザイデル (GS) 法について述べる．GS 法の更新手順は以下のようになる．

$$\boldsymbol{\phi}^{(m+1)} = \boldsymbol{D}^{-1}\left(\boldsymbol{\rho} + \boldsymbol{U}\boldsymbol{\phi}^{(m)} + \boldsymbol{L}\boldsymbol{\phi}^{(m+1)}\right) \tag{2.16}$$

この式を式 (2.15) と同様に式 (2.13) に示した $\boldsymbol{A}$ の要素を用いて成分ごとに書き下すと，以下の式となる．

$$\begin{aligned}\phi_{i,j,k}^{(m+1)} = \frac{1}{d}\Big(&\rho_{i,j,k} + a\cdot\phi_{i,j,k-1}^{(m+1)} + b\cdot\phi_{i,j-1,k}^{(m+1)} + c\cdot\phi_{i-1,j,k}^{(m+1)} \\ &+ c\cdot\phi_{i+1,j,k}^{(m)} + b\cdot\phi_{i,j+1,k}^{(m)} + a\cdot\phi_{i,j,k+1}^{(m)}\Big)\end{aligned} \tag{2.17}$$

Jc 法との相違点は，$\phi_{i,j,k}^{(m+1)}$ の値を計算するために，既に計算されている $\phi_{i,j,k-1}^{(m+1)}$, $\phi_{i,j-1,k}^{(m+1)}$ および $\phi_{i-1,j,k}^{(m+1)}$ の値を使うことである．この相違点により GS 法は Jc 法と比べて同じ反復回数でもより高精度な近似解が得られる（収束性が良い）場合が多い．

これらの反復手順を複数回施すことで近似解を得ることができるが，何回反復計算を行った後の値を近似解とするかを判定する基準が必要である．反復法ではこの指標として残差と呼ばれるベクトルがよく用いられる．m 反復後の残差ベクトル $\boldsymbol{r}^{(m)}$ は以下に示す式で与えられる．

$$\boldsymbol{r}^{(m)} = \boldsymbol{\rho} - \boldsymbol{A}\boldsymbol{\phi}^{(m)} \tag{2.18}$$

具体的な収束判定条件としては，$\boldsymbol{r}^{(0)}$ と $\boldsymbol{r}^{(m)}$ のノルム比 $\|\boldsymbol{r}^{(m)}\|/\|\boldsymbol{r}^{(0)}\|$ である相対残差が十分に小さいことを用いるのが一般的であり，本稿でもユークリッドノルム比 $\|\boldsymbol{r}^{(m)}\|_2/\|\boldsymbol{r}^{(0)}\|_2$ あるいは一様ノルム比 $\|\boldsymbol{r}^{(m)}\|_\infty/\|\boldsymbol{r}^{(0)}\|_\infty$ を，目的に応じて用いる．

2.2.2 ヤコビ法とガウスザイデル法の収束条件と収束性

本節では，Jc 法と GS 法の収束条件，および両者の収束性に関する性質について述べる．本稿で対象とする連立一次方程式 (2.11) に対して係数行列を分解した式 (2.13) を代入し，式 (2.14) と同じ形になるように変形すると，以下の式を得る．

$$\begin{aligned}(\boldsymbol{D}-\boldsymbol{U}-\boldsymbol{L})\,\boldsymbol{\phi} &= \boldsymbol{\rho}\\ \boldsymbol{D}\boldsymbol{\phi} &= \boldsymbol{\rho}+(\boldsymbol{U}+\boldsymbol{L})\,\boldsymbol{\phi}\\ \boldsymbol{\phi} &= \boldsymbol{D}^{-1}\{\boldsymbol{\rho}+(\boldsymbol{U}+\boldsymbol{L})\,\boldsymbol{\phi}\}\end{aligned} \tag{2.19}$$

次に，厳密解 $\boldsymbol{\phi}$ と m 回反復計算を行って得た近似解 $\boldsymbol{\phi}^{(m)}$ との誤差を示す誤差ベクトル $\boldsymbol{o}^{(m)}=\boldsymbol{\phi}-\boldsymbol{\phi}^{(m)}$ を導入し，式 (2.19) から式 (2.14) を引くと以下の式を得る．

$$\begin{aligned}\boldsymbol{o}^{(m+1)} &= \boldsymbol{D}^{-1}\,(\boldsymbol{U}+\boldsymbol{L})\,\boldsymbol{o}^{(m)}\\ &= \boldsymbol{M_{jc}}\boldsymbol{o}^{(m)}\\ &= (\boldsymbol{M_{jc}})^{m+1}\,\boldsymbol{o}^{(0)}\end{aligned} \tag{2.20}$$

式 (2.20) の行列 $\boldsymbol{M_{jc}}=\boldsymbol{D}^{-1}\,(\boldsymbol{U}+\boldsymbol{L})$ は，Jc 法の反復行列と呼ばれる．この反復行列の性質を調べることで，Jc 法の収束条件や性質を知ることができる．式 (2.20) から分かるように m 回反復操作を施した後の誤差ベクトルは，初期誤差ベクトル $\boldsymbol{o}^{(0)}$ に反復行列の m 乗を掛けた形になっている．よって，反復行列の最大の固有値 (スペクトル半径) が 1 未満であることが，Jc 法の収束に関する必要十分条件である．そこで，固有値を求めるために反復行列を以下のように変形する．

$$\begin{aligned}\boldsymbol{M_{jc}} &= \boldsymbol{D}^{-1}\,(\boldsymbol{U}+\boldsymbol{L})\\ -\boldsymbol{D}^{-1}\boldsymbol{D}+\boldsymbol{M_{jc}} &= -\boldsymbol{D}^{-1}\boldsymbol{D}+\boldsymbol{D}^{-1}\,(\boldsymbol{U}+\boldsymbol{L})\\ -\boldsymbol{I}+\boldsymbol{M_{jc}} &= -\boldsymbol{D}^{-1}\,(\boldsymbol{D}-\boldsymbol{U}-\boldsymbol{L})\\ \boldsymbol{M_{jc}} &= \boldsymbol{I}-\boldsymbol{D}^{-1}\boldsymbol{A}\\ \boldsymbol{M_{jc}} &= \boldsymbol{I}-\frac{1}{d}\boldsymbol{A}\end{aligned} \tag{2.21}$$

ここで得た式 (2.21) から，反復行列 $\boldsymbol{M_{jc}}$ の固有値 $\lambda^{\boldsymbol{M_{jc}}}$ を以下のように表すことができる．

$$\lambda^{\boldsymbol{M_{jc}}}=1-\frac{1}{d}\lambda^{\boldsymbol{A}} \tag{2.22}$$

上式は，$\lambda^{\boldsymbol{M}_{jc}}$ を係数行列 $\boldsymbol{A}$ の固有値 $\lambda^{\boldsymbol{A}}$ から算出できることと，両者の固有ベクトルが一致することを示している．そこで $\boldsymbol{A}$ の固有値と固有ベクトルを求める．ここで簡単のために，1 次元ポアソン方程式 $d^2\phi(x)/dx^2 = -\rho(x)$ の固定境界条件から導かれる連立一次方程式を解く問題を一時的に考えると，式 (2.10) に相当する式は以下のように表される．

$$-c\cdot\phi_{i-1} + 2c\cdot\phi_i - c\cdot\phi_{i+1} = \rho_i \tag{2.23}$$

この 1 次元問題の係数行列 $\boldsymbol{A}_{\boldsymbol{x}}$ の固有ベクトルを $\boldsymbol{w}^{\boldsymbol{A}_{\boldsymbol{x}}}$ とし，境界部における値が常に 0 であることを考慮すると，$\boldsymbol{A}_{\boldsymbol{x}}\boldsymbol{w}^{\boldsymbol{A}_{\boldsymbol{x}}} - \lambda^{\boldsymbol{A}_{\boldsymbol{x}}}\boldsymbol{w}^{\boldsymbol{A}_{\boldsymbol{x}}} = 0$ は次のように表される．

$$\begin{aligned}
&-cw_2^{\boldsymbol{A}_{\boldsymbol{x}}} + \left(2c - \lambda^{\boldsymbol{A}_{\boldsymbol{x}}}\right) w_1^{\boldsymbol{A}_{\boldsymbol{x}}} = 0\\
&\qquad\vdots\\
&-cw_{i+1}^{\boldsymbol{A}_{\boldsymbol{x}}} + \left(2c - \lambda^{\boldsymbol{A}_{\boldsymbol{x}}}\right) w_i^{\boldsymbol{A}_{\boldsymbol{x}}} - cw_{i-1}^{\boldsymbol{A}_{\boldsymbol{x}}} = 0\\
&\qquad\vdots\\
&\left(2c - \lambda^{\boldsymbol{A}_{\boldsymbol{x}}}\right) w_{NX-2}^{\boldsymbol{A}_{\boldsymbol{x}}} - cw_{NX-3}^{\boldsymbol{A}_{\boldsymbol{x}}} = 0
\end{aligned} \tag{2.24}$$

これらの式は三項間漸化式とみなすことができ，その特性方程式の解を α と β とすると $w_{i+1}^{\boldsymbol{A}_{\boldsymbol{x}}} + \alpha w_i^{\boldsymbol{A}_{\boldsymbol{x}}} = \beta\left(w_i^{\boldsymbol{A}_{\boldsymbol{x}}} + \alpha w_{i-1}^{\boldsymbol{A}_{\boldsymbol{x}}}\right)$，あるいは $w_{i+1}^{\boldsymbol{A}_{\boldsymbol{x}}} + \beta w_i^{\boldsymbol{A}_{\boldsymbol{x}}} = \alpha\left(w_i^{\boldsymbol{A}_{\boldsymbol{x}}} + \beta w_{i-1}^{\boldsymbol{A}_{\boldsymbol{x}}}\right)$ と表されることから，以下の式が得られる．

$$\begin{aligned}
&\alpha w_{NX-2}^{\boldsymbol{A}_{\boldsymbol{x}}} = \beta^{NX-2} w_1^{\boldsymbol{A}_{\boldsymbol{x}}}, \quad \beta w_{NX-2}^{\boldsymbol{A}_{\boldsymbol{x}}} = \alpha^{NX-2} w_1^{\boldsymbol{A}_{\boldsymbol{x}}}\\
&\left(\frac{\alpha}{\beta}\right)^{NX-1} = 1\\
&\frac{\alpha}{\beta} = \exp\left(\frac{\mathrm{i}\,2l_x\pi}{NX-1}\right) \quad (l_x \in \{0,1,2,3,\ldots,NX-2\})
\end{aligned} \tag{2.25}$$

ただし，$\mathrm{i} = \sqrt{-1}$ とする．これらの式，および特性方程式の解と係数の関係 $\alpha\beta = 1$ から，以下の解を得る．

$$\alpha = \exp\left(\frac{\mathrm{i}\,l_x\pi}{NX-1}\right), \quad \beta = \exp\left(-\frac{\mathrm{i}\,l_x\pi}{NX-1}\right) \tag{2.26}$$

次に，特性方程式の解と係数の関係 $\alpha + \beta = \left(2c - \lambda^{\boldsymbol{A}_{\boldsymbol{x}}}\right)/c$ を用いて l_x に対応する固有値 $\lambda^{\boldsymbol{A}_{\boldsymbol{x}}}(l_x)$ を求め，求まった固有値から固有ベクトル $\boldsymbol{w}^{\boldsymbol{A}_{\boldsymbol{x}}}(l_x)$ を求めると以下のようになる．

$$\lambda^{\boldsymbol{A}_{\boldsymbol{x}}}(l_x) = 4c\sin^2\left(\frac{l_x\pi}{2(NX-1)}\right) \tag{2.27}$$

$$w_i^{\boldsymbol{A}_{\boldsymbol{x}}}(l_x) = \sin\left(\frac{il_x\pi}{NX-1}\right) \tag{2.28}$$

ここで $\boldsymbol{w}^{\boldsymbol{A_x}}(l_x) \neq 0$ から $1 \leq l_x \leq NX-2$ となり，またこれら $NX-2$ 本のベクトルは互いに一次独立であるので，$(NX-2)$ 次元の行列 $\boldsymbol{A_x}$ の全ての固有ベクトルが尽くされている．次に，1 次元問題における Jc 法の反復行列の固有値 $\lambda^{\boldsymbol{M_{jcx}}}$ は，式 (2.22) と式 (2.27) および $d = 2c$ から

$$\lambda^{\boldsymbol{M_{jcx}}}(l_x) = \cos\left(\frac{l_x \pi}{NX-1}\right) \tag{2.29}$$

となる．また，Jc 法の反復行列の固有ベクトルは，係数行列 $\boldsymbol{A_x}$ の固有ベクトルと完全に一致する．

ここで求まった固有値を用いて Jc 法の収束性について述べる．前述のように反復行列におけるスペクトル半径が 1 より小さければその反復法は収束する．対して式 (2.29) の最大値は 1 より小さいことは自明である．よって，ここで想定した連立一次方程式については，Jc 法の収束性が保証される．

次に，ある初期誤差ベクトル $\boldsymbol{o}^{(0)}$ を想定し，Jc 法を用いてこれを 0 に漸近させる，つまり近似解を得ることを考える．まず，任意のベクトルは $\boldsymbol{w}^{\boldsymbol{A_x}}(l_x)$ の一次結合で表現できるため，任意の誤差ベクトル $\boldsymbol{o}^{(m)}$ は以下のように表すことができる．

$$\boldsymbol{o}^{(m)} = \sum_{l_x=1}^{NX-2} \gamma_{l_x}^{(m)} \boldsymbol{w}^{\boldsymbol{A_x}}(l_x) \tag{2.30}$$

なお上式の $\gamma_{l_x}^{(m)}$ は，全ての $\boldsymbol{w}^{\boldsymbol{A_x}}(l_x)$ からなる基底に関する誤差ベクトル $\boldsymbol{o}^{(m)}$ の座標，すなわちこの基底が張る空間での $\boldsymbol{o}^{(m)}$ の座標ベクトル $(\gamma_1^{(m)}, \ldots, \gamma_{NX-2}^{(m)})^{\mathrm{T}}$ の l_x 番目の成分である．そこで以降では，$\gamma_{l_x}^{(m)}$ を $\boldsymbol{w}^{\boldsymbol{A_x}}(l_x)$ に関する誤差成分，あるいは単に波数 l_x の誤差成分と呼ぶ．次に，$\boldsymbol{w}^{\boldsymbol{A_x}}(l_x)$ は反復行列 $\boldsymbol{M_{jcx}}$ の固有ベクトルでもあり，したがって $\boldsymbol{M_{jcx}}\boldsymbol{w}_{\boldsymbol{A_x}}(l_x) = \lambda^{\boldsymbol{M_{jcx}}}(l_x)\boldsymbol{w}_{\boldsymbol{A_x}}(l_x)$ であることから，誤差ベクトル $\boldsymbol{o}^{(m)}$ を以下のように表すことができる．

$$\begin{aligned}\boldsymbol{o}^{(m)} &= \boldsymbol{M}_{\boldsymbol{jcx}}^{m}\boldsymbol{o}^{(0)} = \sum_{l_x=1}^{NX-2} \boldsymbol{M}_{\boldsymbol{jcx}}^{m}\left\{\gamma_{l_x}^{(0)}\boldsymbol{w}_{\boldsymbol{A_x}}(l_x)\right\} \\ &= \sum_{l_x=1}^{NX-2} \gamma_{l_x}^{(0)}\left\{\lambda^{\boldsymbol{M_{jcx}}}(l_x)\right\}^{m}\boldsymbol{w}_{\boldsymbol{A_x}}(l_x) = \sum_{l_x=1}^{NX-2} \gamma_{l_x}^{(m)}\boldsymbol{w}_{\boldsymbol{A_x}}(l_x)\end{aligned} \tag{2.31}$$

式 (2.31) より，波数 l_x の誤差成分は固有値 $\lambda^{\boldsymbol{M_{jcx}}}(l_x)$ の指数関数で減衰すること，すなわち固有値が 0 に近ければ急速に減衰し，1 に近ければ減衰しにくいことがわかる．具体的には，初期誤差ベクトルの成分の中で小さな固有値に対応するもの，すなわち式 (2.29) より波数 l_x が $(NX-1)/2$ に近い誤差成分は急速に減衰し，大きな固有値に対応する $l_x \approx 1$ や $l_x \approx NX-2$ の誤差成分は減衰しにくいことがわかる．実際に，$LenX = NX-1 = 64$, $\boldsymbol{\rho} = 0$ とした 1 次元問題を，初期誤差を $\boldsymbol{o}^{(0)} = \boldsymbol{w}^{\boldsymbol{A_x}}(l_x)$ として Jc 法を用いて求解したときに，収束条件 $\|\boldsymbol{r}^{(m)}\|_2/\|\boldsymbol{r}^{(0)}\|_2 \leq 10^{-2}$ を満たすまでに要する反復回数と波数 l_x との関係は，図 2.2 に示すものとなった．この図からも，波数と収束性の関係を確認することができる．

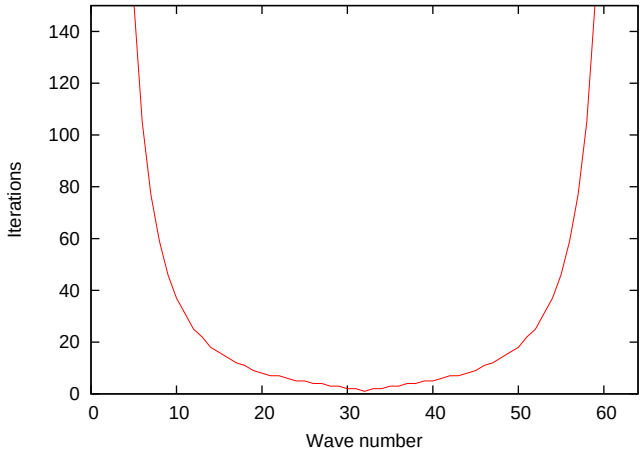


図 2.2: Jc 法による求解での初期誤差の波数と収束までの反復回数の関係

上述のように Jc 法は，波数が大きい誤差成分および小さい誤差成分の収束性に問題がある．そこで下式 (2.32) に示すように，Jc 法の反復式と $\boldsymbol{\phi}^{(m)}$ の重み付きの和を反復式とする重み付きヤコビ法（w-Jc 法）が，波数が大きい誤差成分の収束性を改善した手法としてしばしば用いられている．

$$\boldsymbol{\phi}^{(m+1)} = \omega \boldsymbol{D}^{-1} \{ \boldsymbol{\rho} + (\boldsymbol{U} + \boldsymbol{L}) \boldsymbol{\phi}^{(m)} \} + (1 - \omega) \boldsymbol{\phi}^{(m)} \tag{2.32}$$

また，w-Jc 法の反復行列 $\boldsymbol{M_{\omega jc}}$ は以下のようになる．

$$\boldsymbol{M_{\omega jc}} = \omega \boldsymbol{D}^{-1} (\boldsymbol{U} + \boldsymbol{L}) + (1 - \omega) \boldsymbol{I} = \boldsymbol{I} - \frac{\omega}{d} \boldsymbol{A} \tag{2.33}$$

したがって，$\boldsymbol{M_{\omega jc}}$ の固有値 $\lambda^{\boldsymbol{M_{\omega jc}}}$ は下式 (2.34) により与えられ，1 次元問題に関する固有値 $\lambda^{\boldsymbol{M_{\omega jcx}}}$ は式 (2.35) により与えられる．

$$\lambda^{\boldsymbol{M_{\omega jc}}} = 1 - \frac{\omega}{d} \lambda^{\boldsymbol{A}} \tag{2.34}$$

$$\lambda^{\boldsymbol{M_{\omega jcx}}} (l_x) = 1 - 2\omega \sin^2 \left(\frac{l_x \pi}{2 (NX - 1)} \right) \tag{2.35}$$

w-Jc 法の収束性を保証するためには，任意の l_x に対して

$$|\lambda^{\boldsymbol{M_{\omega jcx}}} (l_x)| < 1 \tag{2.36}$$

を満たす必要があることから，

$$0 < \omega \sin^2 \left(\frac{l_x \pi}{2 (NX - 1)} \right) < 1 \tag{2.37}$$

が導かれ，重み ω に対して $0 < \omega < 1$ の条件が付加される．

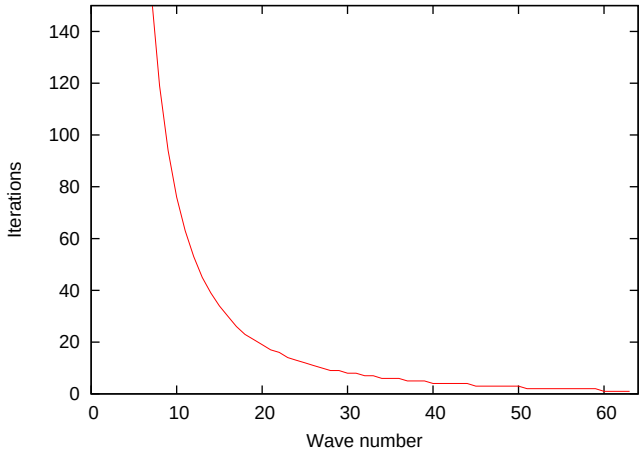


図 2.3: w-Jc 法 ($\omega = 1/2$) による求解での初期誤差の波数と収束までの反復回数の関係

式 (2.35) から，波数が小さい誤差成分に対しては重みの効果は小さいが，波数が大きい誤差成分に対しては効果が強く表れることがわかる．実際，図 2.2 と同じ条件下で，$\omega = 1/2$ とした w-Jc 法の収束に要する反復回数と波数との関係を示した図 2.3 からも，波数が大きい誤差が効果的に減衰していることがわかる．

次に，GS 法の固有値について述べる．Jc 法と同様に GS 法の反復行列 $\boldsymbol{M_{gs}}$ を求めると以下のようになる．

$$\boldsymbol{M_{gs}} = (\boldsymbol{D} - \boldsymbol{L})^{-1} \boldsymbol{U} \tag{2.38}$$

また 1 次元問題の反復行列 $\boldsymbol{M_{gsx}}$ は，実数 R 以下の最大の整数を $\lfloor R \rfloor$，また R 以上の最小の整数を $\lceil R \rceil$ と表記すると，重度が $\lceil (NX-2)/2 \rceil$ のゼロ固有値 $\lambda^{\boldsymbol{M_{gsx}}}(0) = 0$ と，以下に示す $\lfloor (NX-2)/2 \rfloor$ 個の固有値を持つ [11].

$$\lambda^{\boldsymbol{M_{gsx}}}(r_x) = \cos^2\left(\frac{r_x \pi}{NX-1}\right), \quad \left(r_x \in \{1, 2, \ldots, \left\lfloor \frac{NX-2}{2} \right\rfloor\right) \tag{2.39}$$

また，$\lambda^{\boldsymbol{M_{gsx}}}(0)$ の固有ベクトル $\boldsymbol{w}^{\boldsymbol{M_{gsx}}}(0)$ は $(1, 0, \ldots, 0)^{\mathrm{T}}$ であり，$\lambda^{\boldsymbol{M_{gsx}}}(r_x)$ の固有ベクトル $\boldsymbol{w}^{\boldsymbol{M_{gsx}}}(r_x)$ は以下のものとなる．

$$w_i^{\boldsymbol{M_{gsx}}}(r_x) = \left\{\cos\left(\frac{r_x \pi}{NX-1}\right)\right\}^{i-1} \sin\left(\frac{(i-1)\, r_x \pi}{NX-1}\right) \tag{2.40}$$

上記のように，GS 法の反復行列のスペクトル半径も Jc 法と同様に 1 未満であることから，1 次元問題に関する GS 法の収束性は保証される．一方，反復行列の固有ベクトルは式 (2.28) に示した $\boldsymbol{w}^{\boldsymbol{A_x}}(l_x)$ のような正弦波関数の形ではないため，固有ベクトルと固有値を用いた収束性の良否を議論すること

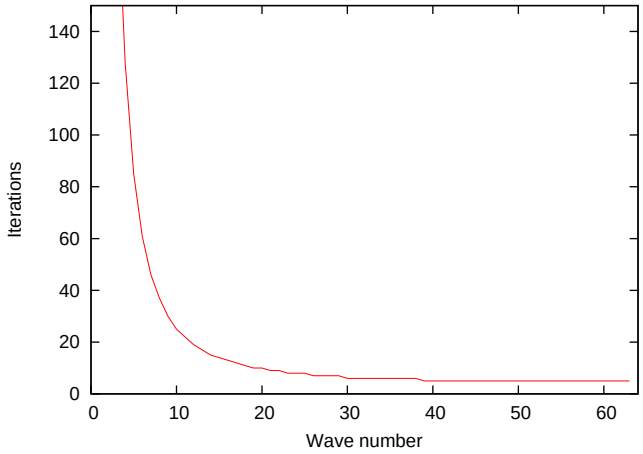


図 2.4: GS 法による求解での初期誤差の波数と収束までの反復回数の関係

は容易ではない．しかし 2.3.3 節でも議論するように，各々の $\boldsymbol{w}^{\boldsymbol{A_x}}(l_x)$ に対する収束性によって反復法の収束性を評価することがしばしば行われている．そこで図 2.2 と同じ条件下で，GS 法の収束に要する反復回数と波数との関係を調べた結果を図 2.4 に示す．この図から，波数が大きい成分は w-Jc 法と同様に急速に減衰することがわかる．また，波数が小さい成分の減衰が遅いことも同様であるが，w-Jc 法に比べれば速く減衰していることもわかる．

最後に，3 次元問題に関する w-Jc 法の反復行列の固有値を求める．まず，x 方向の 1 次元問題における係数行列 $\boldsymbol{A_x}$ と同様に，y 方向における係数行列を $\boldsymbol{A_y}$，z 方向における係数行列を $\boldsymbol{A_z}$ とすると，3 次元問題の係数行列 $\boldsymbol{A}$ は $(NX-2)$ 次単位行列 $\boldsymbol{I_x}$，$(NY-2)$ 次単位行列 $\boldsymbol{I_y}$，$(NZ-2)$ 次単位行列 $\boldsymbol{I_z}$ およびテンソル積を用いて以下のように表される．

$$\boldsymbol{A} = \boldsymbol{I_z} \otimes \boldsymbol{I_y} \otimes \boldsymbol{A_x} + \boldsymbol{I_z} \otimes \boldsymbol{A_y} \otimes \boldsymbol{I_x} + \boldsymbol{A_z} \otimes \boldsymbol{I_y} \otimes \boldsymbol{I_x} \tag{2.41}$$

ここで $\boldsymbol{w}^{\boldsymbol{A_x}}(l_x)$ と同様に，$\boldsymbol{A_y}$ の波数 l_y の固有ベクトルを $\boldsymbol{w}^{\boldsymbol{A_y}}(l_y)$，$\boldsymbol{A_z}$ の波数 l_z の固有ベクトルを $\boldsymbol{w}^{\boldsymbol{A_z}}(l_z)$ として，第 1 項に対して $\boldsymbol{w}^{\boldsymbol{A_z}}(l_z) \otimes \boldsymbol{w}^{\boldsymbol{A_y}}(l_y) \otimes \boldsymbol{w}^{\boldsymbol{A_x}}(l_x)$ を掛けると以下を得る．

$$\begin{aligned}
(\boldsymbol{I_z} \otimes \boldsymbol{I_y} \otimes \boldsymbol{A_x}) &\left\{\boldsymbol{w}^{\boldsymbol{A_z}}(l_z) \otimes \boldsymbol{w}^{\boldsymbol{A_y}}(l_y) \otimes \boldsymbol{w}^{\boldsymbol{A_x}}(l_x)\right\} \\
&= \boldsymbol{w}^{\boldsymbol{A_z}}(l_z) \otimes \boldsymbol{w}^{\boldsymbol{A_y}}(l_y) \otimes \left\{\boldsymbol{A_x}\boldsymbol{w}^{\boldsymbol{A_x}}(l_x)\right\} \\
&= \boldsymbol{w}^{\boldsymbol{A_z}}(l_z) \otimes \boldsymbol{w}^{\boldsymbol{A_y}}(l_y) \otimes \left\{\lambda^{\boldsymbol{A_x}}(l_x)\,\boldsymbol{w}^{\boldsymbol{A_x}}(l_x)\right\} \\
&= \lambda^{\boldsymbol{A_x}}(l_x) \left\{\boldsymbol{w}^{\boldsymbol{A_z}}(l_z) \otimes \boldsymbol{w}^{\boldsymbol{A_y}}(l_y) \otimes \boldsymbol{w}^{\boldsymbol{A_x}}(l_x)\right\}
\end{aligned} \tag{2.42}$$

また第 2 項と第 3 項も同様であるので，行列 $\boldsymbol{A}$ の固有値 $\lambda^{\boldsymbol{A}}(l_x, l_y, l_z)$ と固有ベクトル $\boldsymbol{w}^{\boldsymbol{A}}(l_x, l_y, l_z)$ は，$\boldsymbol{A_y}$ の波数 l_y に対する固有値を $\lambda^{\boldsymbol{A_y}}(l_y)$，$\boldsymbol{A_z}$

の波数 l_z に対する固有値を $\lambda^{\boldsymbol{A_z}}(l_z)$ として，以下のように表される．

$$\boldsymbol{A}\boldsymbol{w}^{\boldsymbol{A}}(l_x,l_y,l_z) = \left\{\lambda^{\boldsymbol{A_x}}(l_x)+\lambda^{\boldsymbol{A_y}}(l_y)+\lambda^{\boldsymbol{A_z}}(l_z)\right\}\left\{\boldsymbol{w}^{\boldsymbol{A_z}}(l_z)\otimes\boldsymbol{w}^{\boldsymbol{A_y}}(l_y)\otimes\boldsymbol{w}^{\boldsymbol{A_x}}(l_x)\right\}$$

$$\lambda^{\boldsymbol{A}}(l_x,l_y,l_z) = \lambda^{\boldsymbol{A_x}}(l_x)+\lambda^{\boldsymbol{A_y}}(l_y)+\lambda^{\boldsymbol{A_z}}(l_z) \tag{2.43}$$

$$\boldsymbol{w}^{\boldsymbol{A}}(l_x,l_y,l_z) = \boldsymbol{w}^{\boldsymbol{A_z}}(l_z)\otimes\boldsymbol{w}^{\boldsymbol{A_y}}(l_y)\otimes\boldsymbol{w}^{\boldsymbol{A_x}}(l_x) \tag{2.44}$$

したがって，3 次元問題に関する w-Jc 法の反復行列の固有値 $\lambda^{\boldsymbol{M_{wjc}}}(l_x,l_y,l_z)$ は

$$\begin{aligned}\lambda^{\boldsymbol{M_{wjc}}}(l_x,l_y,l_z) &= 1-\frac{\omega}{d}\left\{\lambda^{\boldsymbol{A_x}}(l_x)+\lambda^{\boldsymbol{A_y}}(l_y)+\lambda^{\boldsymbol{A_z}}(l_z)\right\}\\ &= 1-\omega\left\{1-\frac{2c}{d}\cos\left(\frac{l_x\pi}{NX-1}\right)-\frac{2b}{d}\cos\left(\frac{l_y\pi}{NY-1}\right)\right.\\ &\qquad\left.-\frac{2a}{d}\cos\left(\frac{l_z\pi}{NZ-1}\right)\right\}\end{aligned} \tag{2.45}$$

と表される．この結果から 1 次元問題と同様に，波数 l_x, l_y, l_z が小さな初期誤差成分は，w-Jc 法の反復の過程で減衰しにくいことがわかる．また GS 法についても，1 次元問題の考察・実験から同様の傾向を示すものと考えられる．

2.3 幾何マルチグリッド法

前節で述べたように，GS 法や w-Jc 法では，波数が小さい $\boldsymbol{w}^{\boldsymbol{A}}(l_x,l_y,l_z)$ に関する誤差成分は減衰しにくい性質がある．幾何マルチグリッド法 [12] は，元の格子空間では減衰しにくい誤差成分を粗い格子空間に幾何的構造にもとづいて写像し，それらをより効率的に減衰させるようにする手法である．具体的には，元の格子空間よりも格子間隔が大きい粗な格子空間を用意し，元の密な空間の問題を粗な空間に写像して求解する．このとき，波数が小さい $\boldsymbol{w}^{\boldsymbol{A}}(l_x,l_y,l_z)$ が相対的に波数の大きなベクトルに変換されることから，これらに関する誤差成分は粗な空間の求解過程で効率的に減衰することが期待できる．元の密な空間については，GS 法や w-Jc 法を用いることで，波数が大きい誤差成分を容易に減衰させることができるため，全体の求解過程を効率化することができる．本節では，このマルチグリッド法について詳細に述べる [13].

2.3.1 2 段グリッド法

はじめに，マルチグリッド法の基礎となる 2 段グリッド法について述べる．解くべき方程式 (2.11) に対して，w-Jc 法または GS 法の反復を pr 回行って，

ある近似解 $\boldsymbol{\phi}^{(pr)}$ を得たとする．このときの誤差ベクトル $\boldsymbol{o}^{(pr)}$ と残差ベクトル $\boldsymbol{r}^{(pr)}$ の関係は，以下に示す誤差方程式により表現される．

$$\begin{aligned}\boldsymbol{A}\boldsymbol{o}^{(pr)} &= \boldsymbol{A}\left(\boldsymbol{\phi}-\boldsymbol{\phi}^{(pr)}\right)\\ &= \boldsymbol{A}\boldsymbol{\phi}-\boldsymbol{A}\boldsymbol{\phi}^{(pr)}\\ &= \boldsymbol{\rho}-\boldsymbol{A}\boldsymbol{\phi}^{(pr)}\\ &= \boldsymbol{r}^{(pr)}\end{aligned} \tag{2.46}$$

$\boldsymbol{\phi}=\boldsymbol{\phi}^{(pr)}+\boldsymbol{o}^{(pr)}$ であるため，ここで求めた誤差方程式 $\boldsymbol{A}\boldsymbol{o}^{(pr)}=\boldsymbol{r}^{(pr)}$ を解くことができれば，求まった誤差ベクトル $\boldsymbol{o}^{(pr)}$ を用いて近似解ベクトル $\boldsymbol{\phi}^{(pr)}$ を修正し，真の解 $\boldsymbol{\phi}$ を得ることができる．よって，効率的にこの誤差ベクトルを求めることを考える．

初期誤差ベクトル $\boldsymbol{o}^{(0)}$ に含まれる誤差成分のうち，波数が大きいものは反復法を用いて減衰させることが容易であるため，ここでは pr 回の反復法でこれらは十分に減衰したとする．したがって，誤差方程式で解くべき誤差ベクトルでは，波数が小さい誤差成分が優位である．そこで，元の空間 $\Omega^{\{h\}}$ と同じ大きさ $LenX\times LenY\times LenZ$ を持ち，各方向の格子間隔がそれぞれ 2 倍であるような粗い格子空間 $\Omega^{\{2h\}}$ に誤差方程式を写像することを考える．この空間では各方向の格子点数がほぼ 1/2 になっているため[2]，各方向の波数 l_x，l_y，l_z のいずれかが格子点数の 1/2 よりも大きな誤差成分は表現することができない．しかし，波数が小さい誤差成分が優位である点を考慮すると，粗い格子空間 $\Omega^{\{2h\}}$ に写像された誤差方程式を解いて得た誤差ベクトルを，空間 $\Omega^{\{h\}}$ に写像して近似解ベクトルを修正すれば，$\Omega^{\{h\}}$ での誤差を大幅に減衰できるものと期待される．

よって，元の空間 $\Omega^{\{h\}}$ での w-Jc 法または GS 法の数回の反復，格子点数が 1/8 の空間 $\Omega^{\{2h\}}$ での連立一次方程式の求解，および誤差ベクトルを用いた近似解の修正という手順で，高い精度の近似解を得ることが期待できる．この 2 段グリッド法と呼ばれる手法の，具体的な計算手順を以下に示す．なお，$\Omega^{\{h\}}$ での係数行列を $\boldsymbol{A}^{\{h\}}$，残差ベクトルを $\boldsymbol{r}^{\{h\}}$ とし，$\Omega^{\{2h\}}$ での係数行列を $\boldsymbol{A}^{\{2h\}}$，残差ベクトルを $\boldsymbol{r}^{\{2h\}}$ とする．また，$\boldsymbol{I}_h^{2h}$ を $\Omega^{\{h\}}$ 上のベクトルを $\Omega^{\{2h\}}$ 上に，$\boldsymbol{I}_{2h}^{h}$ を $\Omega^{\{2h\}}$ 上のベクトルを $\Omega^{\{h\}}$ 上に，それぞれ写像する行列とする．また，式 (2.12) で示した係数行列の非零要素 a，b，c，d を係数行列の表記 $\boldsymbol{A}^{\{h\}}$ に伴って $a^{\{h\}}$，$b^{\{h\}}$，$c^{\{h\}}$，$d^{\{h\}}$ と表記するとすると，$\boldsymbol{A}^{\{2h\}}$ の非零要素 $a^{\{2h\}}$，$b^{\{2h\}}$，$c^{\{2h\}}$，$d^{\{2h\}}$，は格子間隔が 2 倍になっているためそれぞれ $a^{\{2h\}}=1/(2\Delta z)^2=a^{\{h\}}/4$，$b^{\{2h\}}=1/(2\Delta y)^2=b^{\{h\}}/4$，$c^{\{2h\}}=1/(2\Delta x)^2=c^{\{h\}}/4$，$d^{\{2h\}}=d^{\{h\}}/4$ となる．

[2]正確には，たとえば $\Omega^{\{2h\}}$ の x 方向の格子点数は $(NX-1)/2+1$ となる．

$\boldsymbol{A}^{\{h\}}\boldsymbol{\phi} = \boldsymbol{\rho}$ に基づく反復法を

初期解に対して pr 回適用し $\boldsymbol{\phi}^{(pr)}$ を得る (2.47)

$\boldsymbol{r}^{\{h\}} = \boldsymbol{\rho} - \boldsymbol{A}^{\{h\}}\boldsymbol{\phi}^{(pr)}$　残差を計算する (2.48)

$\boldsymbol{r}^{\{2h\}} = \boldsymbol{I}_h^{2h}\boldsymbol{r}^{\{h\}}$　$\Omega^{\{h\}}$ での残差を $\Omega^{\{2h\}}$ に写像する (2.49)

$\boldsymbol{A}^{\{2h\}}\boldsymbol{o}^{\{2h\}} = \boldsymbol{r}^{\{2h\}}$　$\Omega^{\{2h\}}$ での誤差方程式の解を求める (2.50)

$\boldsymbol{o}^{\{h\}} = \boldsymbol{I}_{2h}^{h}\boldsymbol{o}^{\{2h\}}$　$\Omega^{\{2h\}}$ での誤差ベクトルを $\Omega^{\{h\}}$ に写像する (2.51)

$\tilde{\boldsymbol{\phi}} = \boldsymbol{\phi}^{(pr)} + \boldsymbol{o}^{\{h\}}$　誤差ベクトルを用いて $\boldsymbol{\phi}^{(pr)}$ を修正する (2.52)

$\boldsymbol{A}^{\{h\}}\boldsymbol{\phi} = \boldsymbol{\rho}$ に基づく反復法を $\tilde{\boldsymbol{\phi}}$ に po 回適用し $\boldsymbol{\phi}^{(po)}$ を得る (2.53)

上の計算手順の手順 (2.47) や (2.53) のように反復法を数回適用する操作を，マルチグリッド法では波数が大きい誤差成分を減衰させることを目的としているためスムージングと呼び，手順 (2.47) でのスムージングをプリスムージング，手順 (2.53) でのスムージングをポストスムージングと呼ぶ．プリスムージングは初期の誤差ベクトルに含まれる波数が大きい誤差成分を減衰させるために行われるのに対して，ポストスムージングは主に $\Omega^{\{2h\}}$ から $\Omega^{\{h\}}$ への写像で混入する誤差成分の減衰を目的としている．なお，ここで用いられる反復法をスムーザと呼ぶ．また，手順 (2.49) のように格子の密な空間からより粗な空間にベクトルを写像する操作を制約，手順 (2.51) のような制約の逆の操作を補間と呼ぶ．

2 段グリッド法では $\Omega^{\{2h\}}$ を用いることにより，元の空間 $\Omega^{\{h\}}$ では減衰しにくい波数が小さな誤差成分を，少ない計算量で求めることができる．これにより，たとえば $\Omega^{\{2h\}}$ での求解に $\Omega^{\{h\}}$ のスムーザと同じ反復法を用いる場合であっても，格子空間が 1/8 に縮小されていることと，誤差成分の波数が相対的に大きくなって減衰が速まることから，$\Omega^{\{h\}}$ だけを用いた反復求解よりも，効率的に近似解を求めることができる．

2.3.2　V サイクルマルチグリッド法

2.3.1 節では 2 段グリッド法について述べた．本節では，より実用的なマルチグリッド法の一つである V サイクルマルチグリッド法について述べる．2 段グリッド法では，用いる格子が $\Omega^{\{h\}}$ と $\Omega^{\{2h\}}$ の 2 段だけであるため，$\Omega^{\{h\}}$ の格子数が大きい場合，$\Omega^{\{2h\}}$ で解くべき誤差方程式は直接法での求解には依然として非現実的な大きさとなる．また，$\Omega^{\{2h\}}$ での誤差方程式の求解に反復法を用いた場合，$\Omega^{\{2h\}}$ でも波数が小さい誤差成分は減衰しにくいため，十分な精度を持つ近似解を得るためには多数の反復が必要となる．そこで，先の 2 段グリッド法を再帰的に適用し，より多くの粗な格子空間を用意する手法が一般に用いられる．ここで，レベル n の空間を $\Omega^{\{nh\}}$ と表記するとし，最

も粗な格子空間を $\Omega^{\{Lh\}}$ とする L 段マルチグリッド法の計算手順を以下に示す．ただし，$\Omega^{\{nh\}}$ での係数行列，右辺ベクトル，解ベクトル，残差ベクトル，誤差ベクトルをそれぞれ，$\boldsymbol{A}^{\{nh\}}$，$\boldsymbol{\rho}^{\{nh\}}$，$\boldsymbol{\phi}^{\{nh\}}$，$\boldsymbol{r}^{\{nh\}}$，$\boldsymbol{o}^{\{nh\}}$ とし，係数行列 $\boldsymbol{A}^{\{nh\}}$ の非零要素の値をそれぞれ $a^{\{nh\}} = a^{\{h\}}/n^2$，$b^{\{nh\}} = b^{\{h\}}/n^2$，$c^{\{nh\}} = c^{\{h\}}/n^2$，$d^{\{nh\}} = d^{\{h\}}/n^2$ とする．また，$\boldsymbol{A}^{\{nh\}}\boldsymbol{\phi}^{\{nh\}} = \boldsymbol{\rho}^{\{nh\}}$ に反復法を pr 回適用して得た近似解を $\boldsymbol{\phi}^{\{nh\}(pr)}$ とし，$\Omega^{\{nh\}}$ と $\Omega^{\{(n+1)h\}}$ の間の制約行列を $\boldsymbol{I}_{nh}^{(n+1)h}$，補間行列を $\boldsymbol{I}_{(n+1)h}^{nh}$ と表記する

$\boldsymbol{A}^{\{h\}}\boldsymbol{\phi}^{\{h\}} = \boldsymbol{\rho}^{\{h\}}$ に基づく反復法を初期解に pr 回適用し $\boldsymbol{\phi}^{\{h\}(pr)}$ を得る
$\boldsymbol{r}^{\{h\}} = \boldsymbol{\rho}^{\{h\}} - \boldsymbol{A}^{\{h\}}\boldsymbol{\phi}^{\{h\}(pr)}$　残差を計算する
　$\boldsymbol{\rho}^{\{2h\}} = \boldsymbol{I}_h^{2h}\boldsymbol{r}^{\{h\}}$　$\Omega^{\{h\}}$ での残差を $\Omega^{\{2h\}}$ に写像する
　$\boldsymbol{A}^{\{2h\}}\boldsymbol{\phi}^{\{2h\}} = \boldsymbol{\rho}^{\{2h\}}$ に基づく反復法を初期解 (多くの場合は 0 ベクトル)
　　に pr 回適用し $\boldsymbol{\phi}^{\{2h\}(pr)}$ を得る
　$\boldsymbol{r}^{\{2h\}} = \boldsymbol{\rho}^{\{2h\}} - \boldsymbol{A}^{\{2h\}}\boldsymbol{\phi}^{\{2h\}(pr)}$　残差を計算する
　　$\boldsymbol{\rho}^{\{3h\}} = \boldsymbol{I}_{2h}^{3h}\boldsymbol{r}^{\{2h\}}$　$\Omega^{\{2h\}}$ での残差を $\Omega^{\{3h\}}$ に写像する
　　$\vdots$
　　$\boldsymbol{A}^{\{Lh\}}\boldsymbol{\phi}^{\{Lh\}} = \boldsymbol{\rho}^{\{Lh\}}$　最粗格子空間での解を求める
　　$\vdots$
　$\boldsymbol{o}^{\{2h\}} = \boldsymbol{I}_{3h}^{2h}\boldsymbol{\phi}^{\{3h\}(po)}$　$\Omega^{\{3h\}}$ での誤差ベクトルを $\Omega^{\{2h\}}$ に写像する
　$\tilde{\boldsymbol{\phi}}^{\{2h\}} = \boldsymbol{\phi}^{\{2h\}(pr)} + \boldsymbol{o}^{\{2h\}}$　誤差ベクトルを用いて $\boldsymbol{\phi}^{\{2h\}(pr)}$ を修正する
　$\boldsymbol{A}^{\{2h\}}\boldsymbol{\phi}^{\{2h\}} = \boldsymbol{\rho}^{\{2h\}}$ に基づく反復法を $\tilde{\boldsymbol{\phi}}^{\{2h\}}$ に
　　po 回適用し $\boldsymbol{\phi}^{\{2h\}(po)}$ を得る
$\boldsymbol{o}^{\{h\}} = \boldsymbol{I}_{2h}^{h}\boldsymbol{\phi}^{\{2h\}(po)}$　$\Omega^{\{2h\}}$ での誤差ベクトルを $\Omega^{\{h\}}$ に写像する
$\tilde{\boldsymbol{\phi}}^{\{h\}} = \boldsymbol{\phi}^{\{h\}(pr)} + \boldsymbol{o}^{\{h\}}$　誤差ベクトルを用いて $\boldsymbol{\phi}^{\{h\}(pr)}$ を修正する
$\boldsymbol{A}^{\{h\}}\boldsymbol{\phi}^{\{h\}} = \boldsymbol{\rho}^{\{h\}}$ に基づく反復法を $\tilde{\boldsymbol{\phi}}^{\{h\}}$ に po 回適用し $\boldsymbol{\phi}^{\{h\}(po)}$ を得る

上述の計算手順をマルチグリッド法での 1 反復 とし，これを期待する精度の近似解が得られるまで繰り返す．このマルチグリッド法の計算手順で，$\Omega^{\{h\}}$ から $\Omega^{\{2h\}}$ への制約からこの逆の補間までの部分を Coarse Grid Correction という．本稿で述べる V サイクルマルチグリッド法では，図 2.5 に示すように $\Omega^{\{h\}}$ から $\Omega^{\{Lh\}}$ までを順に推移し，その後逆に $\Omega^{\{Lh\}}$ から $\Omega^{\{h\}}$ まで推移するが，たとえば各レベルでの V サイクルを 2 回繰り返す W サイクル法 [14] など，粗な空間の推移方法にはいくつかのバリエーションが存在する．

次に，本稿で議論するマルチグリッド法ソルバでの制約演算と補間演算について述べる．レベル n の格子空間 $\Omega^{\{nh\}}$ の格子点座標 i，j，k に対してレ

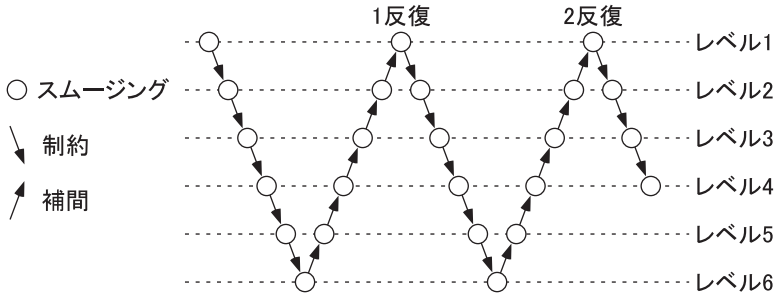


図 2.5: V サイクルマルチグリッド法

ベル $n+1$ の格子空間 $\Omega^{\{(n+1)h\}}$ の格子点座標を I，J，K とし，各段の格子点座標は $i=2I-1$，$j=2J-1$，$k=2K-1$ の関係を持っているとし，レベル n での格子点 番号に対する辞書式順序付けに基づくベクトルまたは行列での要素番号を

$$lex^{\{n\}}(i,j,k) = i-1+\left(\frac{NX-1}{2^{n-1}}-2\right)\cdot(j-2) + \left(\frac{NY-1}{2^{n-1}}-2\right)\cdot\left(\frac{NX-1}{2^{n-1}}-2\right)\cdot(k-2) \quad (2.54)$$

とすると，制約行列 $\boldsymbol{I}_{nh}^{(n+1)h}$ は $l=lex^{\{n+1\}}(I,J,K)$ として以下のように定義される．

$$\left(\boldsymbol{I}_{nh}^{(n+1)h}\right)_{l,m} = \begin{cases} 1/64 & m = lex^n(i\pm1, j\pm1, k\pm1) \\ 1/32 & m \in \{lex^n(i\pm1, j\pm1, k),\ lex^n(i\pm1, j, k\pm1), \\ & \qquad lex^n(i, j\pm1, k\pm1)\} \\ 1/16 & m \in \{lex^n(i\pm1, j, k),\ lex^n(i, j\pm1, k), \\ & \qquad lex^n(i, j, k\pm1)\} \\ 1/8 & m = lex^n(i,j,k) \\ 0 & \text{otherwise} \end{cases} \quad (2.55)$$

この制約行列による $\boldsymbol{r}^{\{nh\}}$ から $\boldsymbol{\rho}^{\{(n+1)h\}}$ への写像を，$\boldsymbol{\rho}^{\{(n+1)h\}}$ の要素ごと

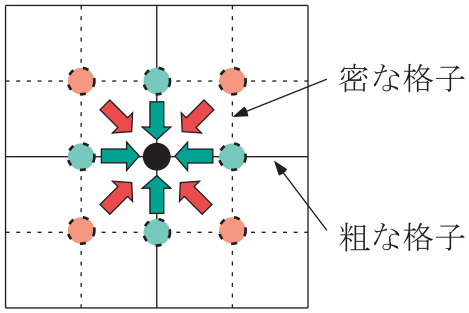


図 2.6: 2 次元問題の制約演算

に書き下すと下式に示すものとなる．

$$\begin{aligned}\rho_{I,J,K}^{\{(n+1)h\}} = \frac{1}{64}\Big(& r_{i-1,j-1,k-1}^{\{nh\}} + 2r_{i,j-1,k-1}^{\{nh\}} + r_{i+1,j-1,k-1}^{\{nh\}} \\ & + 2r_{i-1,j,k-1}^{\{nh\}} + 4r_{i,j,k-1}^{\{nh\}} + 2r_{i+1,j,k-1}^{\{nh\}} \\ & + r_{i-1,j+1,k-1}^{\{nh\}} + 2r_{i,j+1,k-1}^{\{nh\}} + r_{i+1,j+1,k-1}^{\{nh\}} \\ & + 2r_{i-1,j-1,k}^{\{nh\}} + 4r_{i,j-1,k}^{\{nh\}} + 2r_{i+1,j-1,k}^{\{nh\}} \\ & + 4r_{i-1,j,k}^{\{nh\}} + 8r_{i,j,k}^{\{nh\}} + 4r_{i+1,j,k}^{\{nh\}} \\ & + 2r_{i-1,j+1,k}^{\{nh\}} + 4r_{i,j+1,k}^{\{nh\}} + 2r_{i+1,j+1,k}^{\{nh\}} \\ & + r_{i-1,j-1,k+1}^{\{nh\}} + 2r_{i,j-1,k+1}^{\{nh\}} + r_{i+1,j-1,k+1}^{\{nh\}} \\ & + 2r_{i-1,j,k+1}^{\{nh\}} + 4r_{i,j,k+1}^{\{nh\}} + 2r_{i+1,j,k+1}^{\{nh\}} \\ & + r_{i-1,j+1,k+1}^{\{nh\}} + 2r_{i,j+1,k+1}^{\{nh\}} + r_{i+1,j+1,k+1}^{\{nh\}}\Big)\end{aligned} \tag{2.56}$$

すなわちこの制約演算は，格子点 (I,J,K) との距離に応じて，(i,j,k) およびその周辺の格子点に 1/8, 1/16, 1/32 または 1/64 の重みを付与した，残差の重み付き平均を求める演算となっている (図 2.6 参照)．

次に補間演算について述べる．$\Omega^{\{nh\}}$ と $\Omega^{\{(n+1)h\}}$ の格子点座標 (i,j,k) と (I,J,K) との関係は前述のものであるとし，また $m = lex^{\{n+1\}}(I,J,K)$ とすると，補間行列 $\boldsymbol{I}_{(n+1)h}^{nh}$ は以下のように定義される．

$$\left(\boldsymbol{I}_{(n+1)h}^{nh}\right)_{l,m} = \begin{cases} 1/8 & l = lex^n(i\pm1, j\pm1, k\pm1) \\ 1/4 & l \in \{lex^n(i\pm1, j\pm1, k),\ lex^n(i\pm1, j, k\pm1), \\ & \qquad lex^n(i, j\pm1, k\pm1)\} \\ 1/2 & l \in \{lex^n(i\pm1, j, k),\ lex^n(i, j\pm1, k), \\ & \qquad lex^n(i, j, k\pm1)\} \\ 1 & l = lex^n(i,j,k) \\ 0 & \text{otherwise} \end{cases} \tag{2.57}$$

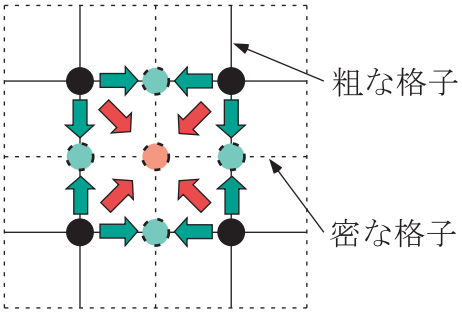


図 2.7: 2 次元問題の補間演算

この補間行列による $\boldsymbol{\phi}^{\{(n+1)h\}}$ から $\boldsymbol{o}^{\{nh\}}$ への写像は，下式に示すように $\boldsymbol{o}^{\{nh\}}$ の格子点座標により 8 つのパターンに分けられる

$$
\begin{aligned}
o^{\{nh\}}_{i,j,k} &= \phi^{\{(n+1)h\}}_{I,J,K} \\
o^{\{nh\}}_{i+1,j,k} &= \frac{1}{2}\left(\phi^{\{(n+1)h\}}_{I,J,K} + \phi^{\{(n+1)h\}}_{I+1,J,K}\right) \\
o^{\{nh\}}_{i,j+1,k} &= \frac{1}{2}\left(\phi^{\{(n+1)h\}}_{I,J,K} + \phi^{\{(n+1)h\}}_{I,J+1,K}\right) \\
o^{\{nh\}}_{i,j,k+1} &= \frac{1}{2}\left(\phi^{\{(n+1)h\}}_{I,J,K} + \phi^{\{(n+1)h\}}_{I,J,K+1}\right) \\
o^{\{nh\}}_{i+1,j+1,k} &= \frac{1}{4}\left(\phi^{\{(n+1)h\}}_{I,J,K} + \phi^{\{(n+1)h\}}_{I+1,J,K} + \phi^{\{(n+1)h\}}_{I,J+1,K} + \phi^{\{(n+1)h\}}_{I+1,J+1,K}\right) \\
o^{\{nh\}}_{i+1,j,k+1} &= \frac{1}{4}\left(\phi^{\{(n+1)h\}}_{I,J,K} + \phi^{\{(n+1)h\}}_{I+1,J,K} + \phi^{\{(n+1)h\}}_{I,J,K+1} + \phi^{\{(n+1)h\}}_{I+1,J,K+1}\right) \\
o^{\{nh\}}_{i,j+1,k+1} &= \frac{1}{4}\left(\phi^{\{(n+1)h\}}_{I,J,K} + \phi^{\{(n+1)h\}}_{I,J+1,K} + \phi^{\{(n+1)h\}}_{I,J,K+1} + \phi^{\{(n+1)h\}}_{I,J+1,K+1}\right) \\
o^{\{nh\}}_{i+1,j+1,k+1} &= \frac{1}{8}\left(\phi^{\{(n+1)h\}}_{I,J,K} + \phi^{\{(n+1)h\}}_{I+1,J,K} + \phi^{\{(n+1)h\}}_{I,J+1,K} + \phi^{\{(n+1)h\}}_{I+1,J+1,K}\right. \\
&\qquad \left. + \phi^{\{(n+1)h\}}_{I,J,K+1} + \phi^{\{(n+1)h\}}_{I+1,J,K+1} + \phi^{\{(n+1)h\}}_{I,J+1,K+1} + \phi^{\{(n+1)h\}}_{I+1,J+1,K+1}\right) \qquad (2.58)
\end{aligned}
$$

すなわちこの補間演算は，格子点 (i,j,k) に最も距離が近い格子点の値の平均を求める演算となっている (図 2.7 参照).

V サイクルマルチグリッド法は，前述のように 2 段グリッド法を再帰的に適用した手順となっている．したがって，2 段グリッド法の $\Omega^{\{2h\}}$ での求解を反復法で行う場合には減衰しにくい誤差成分も，再帰的に $\Omega^{\{3h\}}$ で減衰させる，あるいはそこでも減衰しにくいものは $\Omega^{\{4h\}}$ で減衰させる，というように，誤差成分の波数に応じて減衰に適した粗い格子空間を見出すことができる．すなわち，誤差ベクトル $\boldsymbol{o}^{\{h\}}$ の任意の誤差成分の波数は，いずれかの格子空間では相対的に十分に大きな波数となるため，その空間に関するスムージングで急速に減衰すると期待される．このように，V サイクルマルチグリッド法は，2 段グリッド法よりもさらに効率的な求解手法となっている．

表 2.1: w-Jc 法，GS 法およびマルチグリッド法の収束条件を満たすまでの反復回数

	$NX = NY = NZ$		
	65	129	257
w-Jc 法	19039	74666	292229
GS 法	6347	24890	97411
マルチグリッド法	11	11	11

2.3.3 マルチグリッド法の収束性とスムーザの評価

本節ではマルチグリッド法の収束性について述べる．w-Jc 法の収束性は一般に，$\boldsymbol{w}^{\boldsymbol{A}}(1,1,1)$ に関する誤差成分の減衰速度を定める反復行列の固有値 $\lambda^{\boldsymbol{M}_{wjc}}(1,1,1)$ の値に強く相関している．この固有値は，各方向の格子点数が増加するにつれて 1 に漸近するため，格子点数の増加によって収束性が悪化する．また，GS 法でも同様の傾向を示すことが知られている．

一方マルチグリッド法では，レベル数 L を格子点数の対数に応じて，たとえば $L = \lfloor \log_2 \min(NX, NY, NZ) \rfloor$ と定めることにより，一連の Coarse Grid Correction による $(l_x, l_y, l_z) = (1,1,1)$ の誤差成分の減衰効果は，理想的な制約・補間操作のもとでは格子点数によらず一定のものとなる．したがって．収束にまでに要する V サイクルの数は，格子点数には依存しないものと考えられる．

上記の収束性の格子点数に対する依存・非依存に関する考察を確かめるために，$LenX = LenY = LenZ = 1$, $\boldsymbol{\rho} = 0$, $NX = NY = NZ \in \{65, 129, 257\}$ とした 3 次元問題の求解実験を，w-Jc 法，GS 法，およびマルチグリッド法について行った．なお初期誤差は $\boldsymbol{o}^{(0)} = \boldsymbol{w}^{\boldsymbol{A}}(1,1,1)$ とし，収束条件は $\|\boldsymbol{r}^{(m)}\|_2 / \|\boldsymbol{r}^{(0)}\|_2 \leq 10^{-7}$ とした．ただし m は，w-Jc 法と GS 法では反復回数を，マルチグリッド法では V サイクルの数を，それぞれ意味する．また w-Jc 法では $\omega = 1/2$ とし，マルチグリッド法ではスムーザに GS 法を用いて $pr = po = 1$, $L = \log_2(NX - 1)$ とした．表 2.1 は各々の求解法と $NX(= NY = NZ)$ に対する m の値を示したものであり，w-Jc 法と GS 法では格子点数が増加すると収束性が悪化するのに対し，マルチグリッド法の収束性は格子点数に依存しないことが確認できる．またマルチグリッド法の V サイクル 1 回あたりに要するスムージングの演算コスト C_m は，$pr = po = 1$ であるため各レベルでスムージングが 2 回行われることから，GS 法の 1 反

復に要する演算コストを C_g とすると

$$\begin{aligned} C_m &= \left(2 \cdot \sum_{n=1}^{L} \frac{1}{2^{3(n-1)}}\right) \cdot C_g \\ &= \frac{2 \cdot 2^{3(L+1)} - 1}{7 \cdot 2^{3L}} C_g \end{aligned} \tag{2.59}$$

となり，GS 法の反復回数に換算すると，およそ 16/7 回分と見積もられる．したがって，表の V サイクル数 11 は GS 法では約 25 反復に相当し，制約・補間のコストを勘案してもマルチグリッド法は他の 2 手法に対して圧倒的に優位であることが確認できる．

一方，マルチグリッド法の収束性は，スムーザが以下の条件を満たす波数の誤差成分を，いかに効果的に減衰できるかに依存する．

$$\left(\frac{NX-1}{2} \leq l_x\right) \vee \left(\frac{NY-1}{2} \leq l_y\right) \vee \left(\frac{NZ-1}{2} \leq l_z\right) \tag{2.60}$$

この条件は，誤差成分の波数に $\Omega^{\{h\}}$ でしか表現できないもの（たとえば $l_x \geq (NX-1)/2$）が含まれていることを意味し，w-Jc 法のように反復行列の固有ベクトルが $\boldsymbol{w}^{\boldsymbol{A}}(l_x, l_y, l_z)$ である場合には，この条件を満たす波数に対応する反復行列の固有値の中で絶対値が最大のものが，スムーザによる誤差成分の減衰効果の下限を定める．そこで w-Jc 法では，この絶対値最大の固有値をスムージングファクタと呼び，スムーザの良否を定める指標として用いられる．たとえば $LenX = LenY = LenZ$ かつ $NX = NY = NZ = N$ の場合，すなわち 3 次元問題の w-Jc 法の反復行列の固有値を定める式 (2.45) の係数 a, b, c, d が $a = b = c = d/6$ である場合，固有値 $\lambda^{\boldsymbol{M}_{\omega jc}}(l_x, l_y, l_z)$ の条件式 (2.60) の元での最大絶対値が，重み ω に対するスムージングファクタ $S_{\omega jc}(\omega)$ となる．したがって，$\cos\pi/(N-1) = -\cos\{(N-2)\pi\}/(N-1) \approx 1$ と近似できるとすると，たとえば $l_x = (N-1)/2$, $l_y = l_z = 1$ の固有値，あるいは $l_x = l_y = l_z = N-2$ の固有値が，以下のように $S_{\omega jc}(\omega)$ を与える．

$$S_{\omega jc}(\omega) = \begin{cases} 1 - \dfrac{\omega}{3} & \left(0 < \omega < \dfrac{6}{7}\right) \\ 2\omega - 1 & \left(\dfrac{6}{7} \leq \omega \leq 1\right) \end{cases} \tag{2.61}$$

上式から最もスムージングファクタが小さくなる重みは 6/7 となり，そのときのスムージングファクタは 5/7 となる．

一方 GS 法では，反復行列 $\boldsymbol{M}_{\boldsymbol{gs}}$ の固有ベクトルが $\boldsymbol{w}^{\boldsymbol{A}}(l_x, l_y, l_z)$ のような正弦波関数のテンソル積とはならないため，$\boldsymbol{M}_{\boldsymbol{gs}}$ の固有値に基づいてスムージングファクタを求めることは困難である．そこでローカルモード解析 [15] と呼ばれる手法では，固有値の代わりに $\boldsymbol{M}_{\boldsymbol{gs}}\boldsymbol{w}^{\boldsymbol{A}}(l_x, l_y, l_z)$ と $\boldsymbol{w}^{\boldsymbol{A}}(l_x, l_y, l_z)$ のノルム比を用いて，疑似的にスムージングファクタを求める．すなわち

$$g(l_x, l_y, l_z) = \frac{\left\|\boldsymbol{M}_{\boldsymbol{gs}}\boldsymbol{w}^{\boldsymbol{A}}(l_x, l_y, l_z)\right\|_2}{\left\|\boldsymbol{w}^{\boldsymbol{A}}(l_x, l_y, l_z)\right\|_2} \tag{2.62}$$

とした上で，式 (2.60) の条件下での $g(l_x, l_y, l_z)$ の最大値をスムージングファクタとする．この定義は，初期誤差ベクトル $\boldsymbol{o}^{(0)} = \boldsymbol{w}^{\boldsymbol{A}}(l_x, l_y, l_z)$ に対して，暗に

$$\left\|\boldsymbol{o}^{(m)}\right\|_2 = g(l_x, l_y, l_z)^m \left\|\boldsymbol{o}^{(0)}\right\|_2 \tag{2.63}$$

が成り立つことを仮定しているが，w-Jc 法では $g(l_x, l_y, l_z) = |\lambda^{\boldsymbol{M}_{\omega jc}}|$ であるため成立するのに対し，GS 法では $\boldsymbol{M}_{\boldsymbol{gs}}$ の固有ベクトルが $\boldsymbol{w}^{\boldsymbol{A}}(l_x, l_y, l_z)$ ではないため成立しない．しかしこの定義に基づくスムージングファクタがスムーザの良否を示す指標として有用なことが知られており，実際にスムーザの評価にも用いられている [10][12].

一方，第 3 章で議論する並列化可能なスムーザは，本稿で提案するものも含めて，反復行列の性質を解析的に調べることが困難であるため，ローカルモード 解析を用いてもスムージングファクタを求めることは容易ではない．そこで本稿では，ローカルモード 解析の考え方に基づきつつ，以下に示すように簡単な数値実験により近似的に求めたスムージングファクタについて議論する．まず対象のスムーザを 3 次元問題に適用した場合に，式 (2.60) の条件下で最も減衰しにくいと予想される波数の組 $(\tilde{l}_x, \tilde{l}_y, \tilde{l}_z)$ を，対象スムーザの 1 次元問題への適用実験に基づき定める．具体的には，対象スムーザの 1 次元問題での反復行列をそれぞれ $\boldsymbol{M_x}$，$\boldsymbol{M_y}$，$\boldsymbol{M_z}$ として，$(\tilde{l}_x, \tilde{l}_y, \tilde{l}_z)$ を以下のように $\boldsymbol{o}^{(1)}$ と $\boldsymbol{o}^{(0)}$ のノルム比（相対誤差）を最大化する波数として，数値実験に基づいて定める．

$$\tilde{l}_x = \underset{(NX-1)/2 \le l_x \le NX-2}{\arg\max} \left\|\boldsymbol{M_x} \boldsymbol{w}^{\boldsymbol{A_x}}(l_x)\right\|_2 / \left\|\boldsymbol{w}^{\boldsymbol{A_x}}(l_x)\right\|_2 \tag{2.64}$$

$$\tilde{l}_y = \underset{1 \le l_y \le NY-2}{\arg\max} \left\|\boldsymbol{M_y} \boldsymbol{w}^{\boldsymbol{A_y}}(l_y)\right\|_2 / \left\|\boldsymbol{w}^{\boldsymbol{A_y}}(l_y)\right\|_2 \tag{2.65}$$

$$\tilde{l}_z = \underset{1 \le l_z \le NZ-2}{\arg\max} \left\|\boldsymbol{M_z} \boldsymbol{w}^{\boldsymbol{A_z}}(l_z)\right\|_2 / \left\|\boldsymbol{w}^{\boldsymbol{A_z}}(l_z)\right\|_2 \tag{2.66}$$

次に，このように定めた波数の組と 3 次元問題の反復行列 $\boldsymbol{M}$ を用いてスムージングファクタを下式に基づいて数値実験により定める．

$$S\left(\tilde{l}_x, \tilde{l}_y, \tilde{l}_z\right) = \left\{ \frac{\left\|(\boldsymbol{M})^5 \boldsymbol{w}^{\boldsymbol{A}}\left(\tilde{l}_x, \tilde{l}_y, \tilde{l}_z\right)\right\|_2}{\left\|\boldsymbol{w}^{\boldsymbol{A}}\left(\tilde{l}_x, \tilde{l}_y, \tilde{l}_z\right)\right\|_2} \right\}^{\frac{1}{5}} \tag{2.67}$$

上式は，$\boldsymbol{o}^{(0)} = \boldsymbol{w}^{\boldsymbol{A}}(\tilde{l}_x, \tilde{l}_y, \tilde{l}_z)$ に対してスムーザを 5 回適用して得られる $\boldsymbol{o}^{(5)}$ とのノルム比である相対誤差の相乗平均を，スムージングファクタとすることを意味する．ここで，スムーザの適用回数を 5 回としたのは，しばしば生じる第 1 反復での特異的な減衰の過大な評価を避けつつ，かつ収束に必要な V サイクル数を大きく上回ることによる過小評価も避けるためである．

ここで，式 (2.64) から式 (2.67) に基づいて，$LenX = LenY = LenZ = 1$ かつ $NX = NY = NZ = 513$ の 3 次元問題に関する GS 法のスムージング

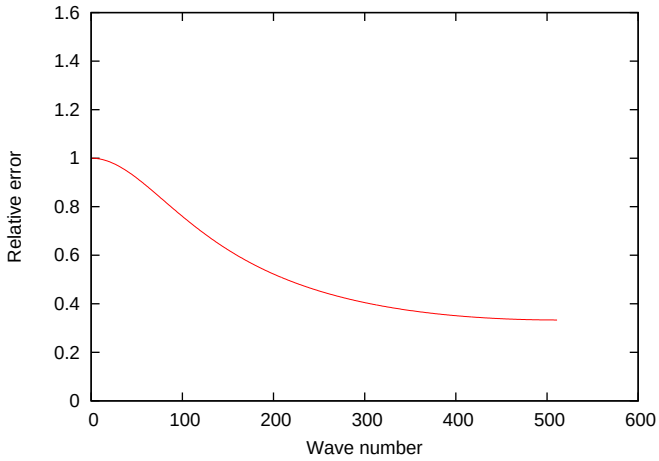


図 2.8: GS 法での初期誤差 $w^A(l_x)$ の波数と相対誤差の関係

ファクタを求める．図 2.8 は数値実験により求めた $\boldsymbol{M_{gsx}}\boldsymbol{w}^{\boldsymbol{A_x}}(l_x)$ と $\boldsymbol{w}^{\boldsymbol{A_x}}(l_x)$ のノルム比と波数 l_x $(1 \leq l_x \leq NX-2=511)$ の関係を示したものであり，この図から $\tilde{l}_x = 256$, $\tilde{l}_y = \tilde{l}_z = 1$ と求めることができる．さらにこの波数を用いた数値実験結果を式 (2.67) に適用することにより，GS 法のスムージングファクタは 0.542 と求められた．この値は w-Jc 法のスムージングファクタ $5/7 \approx 0.714$ よりも小さく，一般に知られている GS 法の優位性を裏付ける結果となっている．

2.4 データの格納形式

本節では，マルチグリッド法の求解プログラムで用いたデータの格納形式について述べる．この節で述べる格納形式は，特に断りがない限り本稿の議論の中で常に維持されているものとする．

まず $\phi_{i,j,k}$ と $\rho_{i,j,k}$ は，大きさが $NX \times NY \times NZ$ である Fortran の 3 次元配列の要素 phi(i,j,k) と rho(i,j,k) に格納され，たとえば GS 法のスムーザは配列 phi(i,j,k) を式 (2.17) に基づいて順次更新するように実装される．また $i \in \{1, NX\}$，$j \in \{1, NY\}$，$k \in \{1, NZ\}$ なる $\phi_{i,j,k}$ は，境界 $\partial\Omega$ に対応するため連立一次方程式の未知変数ではないが，境界に隣接する未知変数 $\phi_{2,2,2}$ などに関する操作を他の変数と共通化するために，常に固定値 0 が格納されるように実装している．すなわち，たとえば $\phi_{2,2,2}$ に関する 1 次方程式は

$$-a\phi_{2,2,3} - b\phi_{2,3,2} - c\phi_{3,2,2} + d\phi_{2,2,2} = \rho_{2,2,2} \tag{2.68}$$

となるが，$\phi_{1,2,2} = \phi_{2,1,2} = \phi_{2,2,1} = 0$ を導入して変形すると，以下のように

一般の $\phi_{i,j,k}$ と同じ形の方程式となる．

$$-a(\phi_{2,2,1}+\phi_{2,2,3})-b(\phi_{2,1,2}+\phi_{2,3,2})-c(\phi_{1,2,2}+\phi_{3,2,2})+d\phi_{2,2,2}=\rho_{2,2,2} \tag{2.69}$$

したがって，たとえば GS 法のスムーザは $2\leq i\leq NX-1$，$2\leq j\leq NY-1$，$2\leq k\leq NZ-1$ なる全ての $\phi_{i,j,k}$ に対して，式 (2.17) の更新操作を i, j, k の値に関わらず実施することとなり，実装が大幅に簡略化される[3]．また粗い格子空間 $\Omega^{\{nh\}}$ に関する $\boldsymbol{\phi}^{\{nh\}}$ および $\boldsymbol{\rho}^{\{nh\}}$ についても，個々の n について $\boldsymbol{\phi}=\boldsymbol{\phi}^{\{h\}}$ や $\boldsymbol{\rho}=\boldsymbol{\rho}^{\{h\}}$ と同様の 3 次元配列を個別に用意する．

次に係数行列の格納形式について述べる．式 (2.10) にあるように $\rho_{i,j,k}$ は $\phi_{i,j,k}$ とその周りの格子点での関数 ϕ の値，つまり $\phi_{i,j,k-1}$，$\phi_{i,j-1,k}$，$\phi_{i-1,j,k}$，$\phi_{i+1,j,k}$，$\phi_{i,j+1,k}$ および $\phi_{i,j,k+1}$ の 7 点の値で記述されている．そこで，式 (2.10) を $\phi_{i,j,k}$ などに関する一次式の特別なものとみなし，各々の係数を 3 次元配列 a，b，c，d，e，f，g の要素として格納する．すなわち，各係数を各々の格子点に固有なものとし，式 (2.10) を以下の一次式の特別なものとして扱えるようにする．

$$\begin{aligned}-a_{i,j,k}\times\phi_{i,j,k-1}-b_{i,j,k}\times\phi_{i,j-1,k}-c_{i,j,k}\times\phi_{i-1,j,k}+d_{i,j,k}\times\phi_{i,j,k}\\ -e_{i,j,k}\times\phi_{i+1,j,k}-f_{i,j,k}\times\phi_{i,j+1,k}-g_{i,j,k}\times\phi_{i,j,k+1}=\rho_{i,j,k}\end{aligned} \tag{2.70}$$

この一般化された一次式に対して，式 (2.10) は各係数を以下のように定めたものとみなすことができる

$$\begin{aligned}a_{i,j,k}&=g_{i,j,k}=-\frac{1}{(\Delta z)^2}\quad(=\text{式 (2.12) の } a)\\ b_{i,j,k}&=f_{i,j,k}=-\frac{1}{(\Delta y)^2}\quad(=\text{式 (2.12) の } b)\\ c_{i,j,k}&=e_{i,j,k}=-\frac{1}{(\Delta x)^2}\quad(=\text{式 (2.12) の } c)\\ d_{i,j,k}&=a_{i,j,k}+b_{i,j,k}+c_{i,j,k}+e_{i,j,k}+f_{i,j,k}+g_{i,j,k}\\ &\qquad\qquad(=\text{式 (2.12) の } d)\end{aligned} \tag{2.71}$$

なお式 (2.70) に基づく一般化により，式 (2.1) で行った単純化，たとえば空間内での拡散係数や格子間隔の固定化などが不適切であるような問題に対しても，本稿で扱う求解プログラムは対応することができる．ただし，第 3 章以降で示す数値実験などは全て式 (2.1) に基づくものであり，また一般化された係数行列の粗な格子空間への写像についても本稿では議論しない．

[3]境界に隣接する未知変数について，固定値 0 を持つ配列要素に対する参照と演算が生じるが，この余分な参照・演算を伴う未知変数は境界部分にしか存在しないため相対的に少量であり，また例外的な取扱いが排除されることによる制御フローの単純化は一般に性能に好影響を与えるため，実装簡略化による性能への悪影響は無視できる程度であると考えられる．

2.5 関連研究

2.5.1 連立一次方程式の反復解法

本節では，ポアソン方程式を離散化して得られる連立一次方程式を対象とした，マルチグリッド法以外の反復解法，具体的には SOR 法 [5]，ADI 法 [16]，および CG 法 [6] についての関連研究を議論する．

文献 [5] で用いている Successive Over-Relaxation(SOR) 法は，Jc 法に対する w-Jc 法と同じ考え方で，GS 法に対して重み付けを付加した手法である．具体的には GS 法の反復式（再掲）

$$\boldsymbol{\phi}^{(m+1)} = \boldsymbol{D}^{-1}(\boldsymbol{\rho} + \boldsymbol{U}\boldsymbol{\phi}^{(m)} + \boldsymbol{L}\boldsymbol{\phi}^{(m+1)}) \tag{2.16}$$

に対して，重み（加速係数）ω を付加した下式が SOR 法の反復式である．

$$\boldsymbol{\phi}^{(m+1)} = \omega\boldsymbol{D}^{-1}(\boldsymbol{\rho} + \boldsymbol{U}\boldsymbol{\phi}^{(m)} + \boldsymbol{L}\boldsymbol{\phi}^{(m+1)}) + (1-\omega)\boldsymbol{\phi}^{(m)}$$

w-Jc 法の重み付の効果と同様に，適切な加速係数 ω を設定することで SOR 法は GS 法よりも優れた収束性を示すことが知られている．ただし 2.3.3 節で議論した w-Jc 法や GS 法と同様に，格子点数が増加すると収束までの反復回数も増加することもやはり知られている．

文献 [16] で述べられている Alternating-Direction Implicit (ADI) 法は，熱伝導方程式 $\partial\phi/\partial t = \nabla^2\phi$ の陰的解法の一種であり，この時間発展を反復して解くことによってポアソン方程式の解が定常状態として得られる．2 次元熱伝導方程式を対象とし，時間刻みを Δt，空間 2 階中心差分演算子を δ_x^2，δ_y^2 としたとき，陰的解法の一つである Crank-Nicolson 法が

$$\frac{\phi(t+\Delta t) - \phi(t)}{\Delta t} = \frac{(\delta_x^2 + \delta_y^2)(\phi(t+\Delta t) + \phi(t))}{2}$$

の形の連立一次方程式を Δt ごとに解くのに対し，ADI 法では下式に示すように x 方向と y 方向の差分を分離して $\Delta t/2$ ごとに解く．

$$\begin{aligned}
\frac{\phi(t+\Delta t/2) - \phi(t)}{\Delta t/2} &= \delta_x^2\phi(t+\Delta t/2) + \delta_y^2\phi(t) \\
\frac{\phi(t+\Delta t) - \phi(t+\Delta t/2)}{\Delta t/2} &= \delta_x^2\phi(t+\Delta t/2) + \delta_y^2\phi(t+\Delta t)
\end{aligned}$$

この結果，上記の 2 式からそれぞれ導かれる連立一次方程式の係数行列はいずれも三重対角行列となり，未知変数の数 N に対して $\mathcal{O}(N)$ の演算量で直接法（Thomas 法）により解くことができる．したがって Crank-Nicolson 法から導かれる連立一次方程式を直接法あるいは反復法で解くよりも計算量が遥かに小さく，また陽的解法と比べて Δt を大きく取ることができる．しかしポアソン方程式の求解の観点では，ADI 法と加速係数が最適な SOR 法の収束性が同等であることが証明されており，Thomas 法が SOR 法よりも演算量

が大きいことを考慮すると SOR 法が有利であると言える．そこで収束性改善のために文献 [17] では，定常状態に近づくに連れて ϕ の変化が小さくなることを利用して，解の精度を保ちつつ Δt を反復ごとに大きくする方法が示されている．具体的には，あらかじめ与えられた反復回数に対して最良の近似解を与える反復ごとの Δt を定める方法と，これによって SOR 法よりも優れた収束性が得られることの証明が，文献 [17] には示されている．ただしこの方法を用いても，格子点数が増加すると収束までの反復回数も増加することが証明されている [18].

文献 [6] で用いられている共役勾配 (Conjugate Gradient : CG) 法は，対称正定値であるような係数行列 $\boldsymbol{A}$ と真の解 $\boldsymbol{\phi}^*$ について，内積 $F(\boldsymbol{\phi}) = (\boldsymbol{\phi} - \boldsymbol{\phi}^*, \boldsymbol{A}(\boldsymbol{\phi} - \boldsymbol{\phi}^*))$ で表現される 2 次形式の最小化問題の唯一解が $\boldsymbol{\phi} = \boldsymbol{\phi}^*$ であることに基づいて，近似解ベクトル $\boldsymbol{\phi}^{(m)}$ の列を定める勾配法の一種である．具体的には探索ベクトルと呼ばれるベクトル列 $\boldsymbol{p}^{(0)} = \boldsymbol{r}^{(0)}$，$\boldsymbol{p}^{(1)}$，... を用いて $\boldsymbol{\phi}^{(m+1)} = \boldsymbol{\phi}^{(m)} + \alpha^{(m)}\boldsymbol{p}^{(m)}$ とした上で，$F(\boldsymbol{\phi}^{(m+1)})$ を最小化するように $\alpha^{(m)} = (\boldsymbol{r}^{(m)}, \boldsymbol{p}^{(m)})/(\boldsymbol{p}^{(m)}, \boldsymbol{A}\boldsymbol{p}^{(m)})$ とする．この探索ベクトルを定める方法はいくつか知られているが [19]，CG 法の特徴は全ての探索ベクトルが $\boldsymbol{A}$ に関して互いに共役となる，すなわち任意の $l \neq m$ について $(\boldsymbol{p}^{(l)}, \boldsymbol{A}\boldsymbol{p}^{(m)}) = 0$ となるように定めることにある．これにより未知変数の数を N とし，また $\Phi^{(m)} = \{\boldsymbol{\phi}^{(0)} + q | q \in \mathrm{span}(\boldsymbol{p}^{(0)}, \boldsymbol{p}^{(1)}, \ldots, \boldsymbol{p}^{(m-1)})\}$ とすると，すべての $m < N$ について

$$\boldsymbol{\phi}^{(m)} = \arg \min_{\phi \in \Phi^{(m)}} F(\phi) \tag{2.72}$$

が成り立つことから，$F(\boldsymbol{\phi}^{(N)}) = 0$ すなわち $\boldsymbol{\phi}^{(N)} = \boldsymbol{\phi}^*$ が保証され，高々N 回の反復で真の解を得ることができる．

しかし N 回以下の反復での求解保証は数学的に厳密な演算を前提としており，浮動小数点演算では収束条件を満たす近似解であっても N 回以上の反復を要することもあることが知られている．また仮に所望の精度を持つ近似解を得るために必要な反復回数が $\mathcal{O}(N)$ であるとすると，求解に要する計算量は $\mathcal{O}(N^2)$ となる．そこで CG 法の収束性向上のため，求解対称の方程式を変形した上で CG 法を適用するさまざまな前処理付き CG(Preconditioned CG : PCG) 法が提案されている．PCG 法では，方程式 $\boldsymbol{A}\boldsymbol{\phi} = \boldsymbol{\rho}$ を二つの前処理行列 $\boldsymbol{C}_1$ と $\boldsymbol{C}_2$ を用いて $(\boldsymbol{C}_1^{-1}\boldsymbol{A}\boldsymbol{C}_2^{-1})(\boldsymbol{C}_2\boldsymbol{\phi}) = \boldsymbol{C}_1^{-1}\boldsymbol{\rho}$ と変形し，これに対して CG 法による求解操作を施す．このとき $(\boldsymbol{C}_1^{-1}\boldsymbol{A}\boldsymbol{C}_2^{-1})$ の最大・最小固有値の比が 1 に近いこと，すなわち $\boldsymbol{C}_1\boldsymbol{C}_2 \approx \boldsymbol{A}$ であることが CG 法の収束性の向上に有効であることと，反復手順の中で必要な $(\boldsymbol{C}_1\boldsymbol{C}_2)\boldsymbol{q}^{(k)} = \boldsymbol{r}^{(k)}$ なる $\boldsymbol{q}^{(k)}$ を求めるコストが小さいことが求められる．

この要求を満たす前処理にはさまざまなものが提案されているが [19]，文献 [20] では w-Jc 法または対称 SOR 法により，また文献 [21] ではマルチグリッド法により，それぞれ $\boldsymbol{A}\boldsymbol{q} = \boldsymbol{r}^{(k)}$ の近似解を求めて $\boldsymbol{q}^{(k)}$ とする方法が

用いられている．これらの方法では，反復法によって定まる反復行列を $\boldsymbol{M}$，反復回数を l，$\boldsymbol{q}$ の初期解を 0 としたとき，$\boldsymbol{C}_1\boldsymbol{C}_2 = (\boldsymbol{I} - (\boldsymbol{M})^l)^{-1}\boldsymbol{A}$ となる．なお，これらの中でマルチグリッド法前処理が CG 法の収束性を大幅に向上させることが知られているが [22]，$\boldsymbol{q}^{(k)}$ を求めるコストが PCG 法の求解コストの大きな部分を占めるため，本稿で議論するような効率的な並列化が有用であることに留意すべきである．

一方文献 [23] では，$\boldsymbol{A}$ を不完全コレスキー (Incomplete Cholesky : IC) 分解によって下三角行列と上三角行列に分解したものを $\boldsymbol{C}_1$ および $\boldsymbol{C}_2$ とする IC 前処理手法が用いられている．IC 分解は，完全なコレスキー分解の結果の $\boldsymbol{A}$ のゼロ要素に対応する要素を強制的に 0 としたものと考えることができ，$\boldsymbol{A}$ が疎行列である場合には所要メモリ容量や分解に要する計算時間を小さく抑えることができる．また $\boldsymbol{r}^{(k)}$ から $\boldsymbol{q}^{(k)}$ を求める処理を，$\boldsymbol{C}_1$ と $\boldsymbol{C}_2$ が三角行列であることを利用した前進代入と後退代入によって $\mathcal{O}(N)$ で実現できるという特徴もある．なお，前進代入・後退代入はいずれも逐次性を持つ処理であるため，2.5.4 節で述べるような彩色順序付けによる並列化手法が提案されている．

2.5.2 マルチグリッド法

本節では一般的なマルチグリッド法について，2.3.2 節で述べた幾何的な V サイクルマルチグリッド法とは異なるものを取り上げて議論する．

V サイクルマルチグリッド法は，2 段グリッド法の $\Omega^{\{2h\}}$ の求解に 2 段グリッド法を適用し，その結果得られる $\Omega^{\{3h\}}$ に対してさらに 2 段グリッド法を適用するという形で，2 段グリッド法の再帰適用をレベルごとに 1 回ずつ行ったものとみなすことができる．この適用回数を一般化して μ 回としたものが，文献 [14] に述べられている μ サイクル法であり，$\mu = 2$ のものは特に W サイクル法と呼ばれている．$\mu = 1$ である V サイクル法に比べ，$\mu > 1$ の場合は粗い格子でのスムージング操作 (Coarse Grid Correction) の回数が増えるため，V サイクル法ではこの効果が不十分であるような問題に対して効果的であるとされている．また，文献 [24] で用いられている (Full Multigrid : FMG) 法では，各レベルに固有の収束条件を設けることができ，これが満たされるまでより下位のレベルを対象とする V サイクルを繰り返す．したがって，まず最粗格子まで降下して条件が満たされるまでスムージングが繰り返され，次に 1 段ずつレベルを上げて条件を満たすまで V サイクルを繰り返す処理が，最密格子まで反復される．μ サイクル法では Coarse Grid Correction の回数や方法があらかじめ定められるのに対し，FMG では問題に応じて適応的に制御できるという特徴がある．

2.3 節で述べた幾何マルチグリッド法は，その名称が示すように問題が持つ幾何学的な性質を利用して粗い格子や格子間の制約・補間演算を定義している．

したがって本稿で扱うような構造格子に基づく問題には適しているが，有限要素法でしばしば用いられる 4 面体や不規則 6 面体などで構成された非構造格子に対しては，粗い格子を幾何学的に構築することが容易ではない．文献 [25] で論じられている代数的マルチグリッド (Algebraic Multigrid : AMG) 法は，これらの非構造格子を含むより一般的な問題にマルチグリッド法を適用するために考案されたもので，係数行列の非対角成分の値に基づき制約・補間演算を定義する点が幾何マルチグリッド法との大きな違いである．古典的な AMG 法では係数行列 $\boldsymbol{A}$ の行 i について，非対角要素の最大絶対値 $\max_{k\neq i}|A_{i,k}|$ を基準として，他の非対角要素 $A_{i,j}$ をその絶対値と閾値 θ $(0<\theta<1)$ により，$|A_{i,j}| \geq \theta \max_{k\neq j}|A_{i,k}|$ である相対的に大きな要素と，そうではない小さな要素に分類する．また列番号 j と i の関係を，$A_{i,j}$ が大きな要素であれば強い接続，小さな要素であれば弱い接続と呼ぶ．スムージングによって減衰しにくい誤差成分 $\boldsymbol{o}$ について，i と j が強い接続である場合には $o_i \approx o_j$ となる性質があるため，粗い格子を生成する際には i と j の一方を間引くことによって Coarse Grid Correction による誤差の減衰を促す．また，粗い格子点には属さない i に関する補間演算を i と強い接続を持つ粗い格子点の重み付きの和とし，重みを係数行列の行 i の要素と i と接続関係を持つ行 j の要素および i と j の接続の強弱により定め，制約行列は補間行列の転置行列とするのが一般的である．このように，係数行列の値に基づいて粗い格子の構成と制約・補間演算を定義するため，AMG 法は非構造格子を用いて離散化した問題に対しても適用可能な手法である．

また文献 [26] では加法的なマルチグリッド法を取り上げ，2.3 節で述べた乗法的なマルチグリッド法，すなわちスムージング後の残差を粗い格子に写像する方法との比較を行っている．加法マルチグリッド法では，直前の V サイクルで得られた残差を全レベルの粗い格子にそれぞれ写像して右辺ベクトルとし，各レベルで独立にスムージングを行った結果を補間演算により最粗格子から順に統合することで，最密格子での近似解の修正を行う．この両者を比較すると，スムージングが各レベルで相補的に影響を及ぼし合う乗法的マルチグリッド法が，Coarse Grid Correction の効果がより強く収束性も良好であると考えられる．実際に文献 [26] では，乗法的なマルチグリッド法が収束性の面で優位であることが示されている．

2.5.3 並列化スムーザ

既存の代表的な並列スムーザである赤-黒順序付け GS (RB-GS) スムーザでは，各格子点を赤・黒の 2 色に塗り分けることで GS 法の並列化を行うが，この塗り分けを直方体状のブロック単位とすることで性能を改善した新たな並列化スムーザであるブロック化赤-黒順序付け GS (BRB-GS) スムーザを，第 3 章で提案する．またこの章では，ブロックに対する逐次 GS スムージング

を複数回実施することでさらに性能を向上させた改良型の mBRB-GS スムーザも提案し，これらと RB-GS スムーザや GS 法と w-Jc 法を組み合わせたハイブリッド (Hybrid) スムーザとの性能比較を行う．そこで本節では，既存の並列スムーザである RB-GS および Hybrid スムーザに関連する研究と， その他の GS スムーザの並列化手法について議論する．

文献 [27] で用いられている RB-GS スムーザは，第 3 章で詳述するように各格子点を同じ色が隣り合わないように赤と黒の 2 色で塗り分けて順序付けることで，同色の格子点の更新計算を並列に実施できるようにする手法である．また差分近似に 7 点以上の格子点が用いられるため赤・黒 2 色では彩色できない場合の一般的な並列化手法として，文献 [28] ではより多くの色を使った多色順序付け法が示されている．RB-GS スムーザは最も利用頻度の高い並列化スムーザであるため，これを発展させた手法や最適化手法が多数提案されている．たとえば文献 [29] には，SOR スムーザに対して赤-黒順序付け法を適用し，逐次的な SOR と同様に適切な加速係数を付与することで，RB-GS スムーザより優れた収束性が得られることが示されている．また文献 [30] では，第 3 章で詳述する RB-GS スムーザの問題点である低い参照局所性を改善するために，格子空間をブロック化してブロック内の赤格子点の更新計算の後に同じブロック内の黒格子点[4]の更新計算を実施する方法が示されている．この方法を用いることで，赤格子点の更新計算で参照・更新した格子点の値がキャッシュに残存している状況で，それらを更新・参照する黒格子点の更新計算が実施されるため，時間的な参照局所性が大幅に改善される．さらに文献 [31] ではこれを発展させ，RB-GS スムージング処理の反復回数が 2 以上である場合に，2 回目以降のスムージングも同じブロックに対して連続的に行う[5]手法が提案されている．

文献 [32] で用いられている Hybrid スムーザは，領域分割法での並列計算単位である部分領域の境界部に w-Jc 法を，また内部には GS 法を用いる方法であり，これを採用した AMG 法ソルバが hypre と呼ばれるライブラリの一部として公開されている [33]．また文献 [34] では，w-Jc 法の代わりに多色順序付け GS 法を用いる一種の改良版が提案されている．

また，上記の手法がいずれも逐次的な GS 法あるいは SOR 法に何らかの変形を加えたものであるのに対し，逐次 GS 法や SOR 法と全く同じ計算を並列に行う手法も提案されている．文献 [35] の Hyperplane Partitioning は，ある格子点 (i, j, k) の計算結果に直接依存する格子点 $(i+1, j, k)$，$(i, j+1, k)$ および $(i, j, k+1)$ に関する計算が，相互に依存関係を持たないため並列化可能であること，またこのことがこれら 3 点が属する平面上の全ての格子点について成り立つことを利用して，SOR 法を並列化する手法である．またこれを発展させた手法として，文献 [36] では Pipelining Partitioning が提案され

[4]厳密には，ブロックを各方向について 1 格子点分ずらしたブロック内の黒格子点が対象となる．

[5]厳密には，やはり対象ブロックが 1 格子点分ずれる．

ている．この方法では，格子空間を z 軸に垂直な平面で T 個に分割し，T 個のスレッド（またはプロセス）t_0, t_1, ..., t_{T-1} に対して z 座標が小さい順に部分領域を与え，各スレッドは与えられた部分領域のスムージングを xz 平面を単位として行う．その際，t_0 による $y = j$ の平面のスムージングと並列に，全ての $1 \leq l < T$ について t_l は $y = j - l$ の平面のスムージングを行い，一つの xz 平面のスムージングが終了するごとに全スレッドで同期を取って次の平面の処理に移行する．この方法も Hyperplane Partitioning と同様に逐次 GS 法と完全に同じ計算を行うが，Hyperplane Partitioning では並列実行可能な平面上の格子点の集合がメモリ空間上で全く隣接しないため，各スレッドが行う更新操作の対象格子点が不連続になるのに対し，Pipelining Partitioning では xz 平面を単位として計算を行うため，x 方向へ連続したアクセスを行うことができる．

以上で述べた並列化手法は，いずれもスレッド並列あるいはプロセス並列実行を対象としたものであるが，第 5 章で述べるようにメニーコアプロセッサ Xeon Phi をはじめ多くのプロセッサに搭載されている SIMD 演算機構を活用するための並列化も重要であり，文献 [37] では RB-GS スムーザを用いたマルチグリッド法の性能を，Xeon Phi を用いて評価している．また文献 [38] では，差分近似の方法を変更することによって SOR 法に対するベクトル計算機でのベクトル並列化を実現しており，これは SIMD 並列化を行う場合でも有効な手法である．具体的には，2 次元の問題に対する一般的な 5 点差分法では $\phi_{i,j}$ に関する空間中心差分を $\phi_{i-1,j}$，$\phi_{i,j-1}$，$\phi_{i+1,j}$，$\phi_{i,j+1}$ の 4 点を用いて表すが，この文献では $\phi_{i-1,j-1}$，$\phi_{i-1,j+1}$，$\phi_{i+1,j-1}$，$\phi_{i+1,j+1}$ の 4 点を用いる．このことにより，$\phi_{i,j}$ と $\phi_{i+1,j}$ の更新操作に関するフロー依存性がなくなるため，x 方向に進行するループの SIMD 並列化が可能となっている．

2.5.4 IC 前処理の並列化手法

第 3 章で提案する BRB-GS スムーザは，格子空間を直方体ブロックを単位として分割し，隣接ブロックが同色とならないように各ブロックを赤または黒に彩色する，ブロック化赤-黒順序付け法（BRB 順序付け法）によって GS スムーザを並列化したものである．この BRB 順序付け法は元来，2.5.1 節で述べた CG 法の IC 前処理の並列化手法として，文献 [39] で提案されたものである．

前述のように IC 前処理では，係数行列 $\boldsymbol{A}$ を IC 分解して得られる下三角行列 $\boldsymbol{C}_1$ と上三角行列 $\boldsymbol{C}_2$ を用いて，$(\boldsymbol{C}_1\boldsymbol{C}_2)\boldsymbol{q}^{(k)} = \boldsymbol{r}^{(k)}$ なる $\boldsymbol{q}^{(k)}$ を前進代入・後退代入により求めるが，これらの操作はいずれも逐次性を内包しているため並列化の阻害要因となる．この前進後退代入の並列化にも，GS 法の並列化と同じ考え方を利用することができ，たとえば文献 [40] では多色順序

付けによる並列化が行われている．ただしこの並列化を行うと，$\boldsymbol{C}_1\boldsymbol{C}_2$ が逐次的な処理とは異なるものとなり，特に色数を最小限に留めると CG 法の収束性に悪影響を及ぼすことが知られている．そこで並列化に必要な数以上の色を使って彩色することで収束性の悪化を防止することが多いが，色数に等しい同期操作が必要になるため計算性能の劣化が問題となる [41]．一方文献 [42] では，2.5.3 節で述べた Hyperplane Partitioning と同じ考え方（文献では Wavefront 法と呼んでいる）によって前進・後退代入を並列化している．たとえば 3 次元ポアソン方程式の有限差分近似で得られる係数行列では，格子点 (i,j,k) に関する $\boldsymbol{C}_1$ の行に出現する非零要素は $(i-1,j,k)$，$(i,j-1,k)$ および $(i,j,k-1)$ のみであり，この性質を用いると $\boldsymbol{C}_1$ を用いた前進代入処理に対して逐次処理と全く同じ結果を得る並列化が可能である．しかし 2.5.3 節で述べたように，並列実行可能な平面上の格子点の集合がメモリ空間上で全く隣接しないという，メモリアクセスに関する問題点がある．

これらに対して BRB 順序付け法による並列化は，赤・黒 2 色であるため同期オーバヘッドが最小化され，並列化の単位をブロックとすることで逐次処理に近い計算順序が維持され，かつブロック内の計算ではメモリアクセスが連続的になるという，多色順序付け法や Hyperplane Partitioning の問題点がほぼ完全に解決された手法となっている．その結果．これらの手法に比べて優れた性能が実現されている．なお，前述のように前進後退代入の並列化と GS 法の並列化には考え方に共通性があることと，一定の逐次性を維持したブロック単位の処理は GS 法に対しても有効と期待されたことが，第 3 章以降で述べる研究の出発点となっている．

第3章　マルチコアプロセッサ向けの高性能な並列化スムーザ

3.1　背景

本章では，マルチグリッド法の共有メモリ環境における並列化について述べる．前章でも述べたように，マルチグリッド法の収束性はスムージングで用いるスムーザの性能によって変化する．また並列化の観点でも，スムーザによってその難易には差がある．たとえば GS スムーザは，w-Jc スムーザに比べて良好な収束性を示すが，近似解ベクトルの参照・更新に要素間の依存関係があるため，要素ごとの演算を単純に並列実行することはできない．そこで従来は，GS 法を基本としつつ部分的に w-Jc 法を組み合わせたハイブリッド (Hybrid) スムーザ [32] や，赤-黒順序付け法に基づいて GS スムーザを並列化する赤-黒順序付け GS (RB-GS) スムーザ [43] が用いられてきた．特に後者は，Hybrid スムーザに見られる並列化に伴う収束性の劣化を生じず，マルチグリッド法によるポアソンソルバに広く用いられている．しかし，RB-GS 法に対して標準的に用いられるストライドアクセスによる実装は，対象空間が自然に表現されることから，内部でポアソン方程式を求解するシミュレーションコードとの親和性が良いが，キャッシュ内のデータ再利用性が低いという問題がある．そこで本章では，並列 3 次元ポアソンソルバの高速化を目的とした，新たな並列化スムージング手法について論じる．

本章ではまず，GS スムーザの並列化手法として，IC/ILU 分解前処理のために提案されたブロック化赤-黒順序付け法 [39][44] を利用した GS スムーザ (BRB-GS スムーザ) を提案する．この順序付け法は，格子空間を複数のブロックに分割し，これらのブロックに対して赤-黒順序付け法を適用するものであり，各ブロック内の配列要素へのアクセスが連続的となる利点がある．次に，BRB-GS スムーザの改良版として，個々のブロックに対する GS スムージングを複数回連続して行う，改良型ブロック化赤-黒順序付け GS スムーザ (mBRB-GS スムーザ) を提案する．このスムーザは通常の GS スムーザとは異なり，乗法シュワルツスムーザの一種とみなすことができる．また，mBRB-GS スムーザの特徴を利用して制約・補間演算とスムージングをキャッシュブロッキングする実装法を提示する．さらに 4 つのマルチコアプロセッサを搭載した SMP ノードを使用して，数値実験による各並列化スムーザの性能を評価し，提案するスムーザによるマルチグリッドポアソンソルバが，従

来の RB-GS スムーザによるものと比べて，2 倍以上高速であることも示す．

3.2 既存の並列化スムーザ

マルチグリッド法の手順はスムージングと残差の計算，制約・補間演算からなる．このうち残差の計算と制約・補間演算は格子点毎に独立であるため，格子空間を分割することにより容易に並列化することができる．一方スムーザは，その種別により並列化の難易度に差がある．たとえば w-Jc スムーザでは，解ベクトルの更新操作に要素間の依存関係が存在しないため，並列化は容易である．これに対して辞書式順序付けに基づく標準的な GS スムーザでは，あるインデックスを持つ要素の更新操作において，それよりも小さなインデックスを持つ要素の更新後の値が参照されるため，この依存関係による順序を保ちつつ並列化することは容易ではない．しかし，GS スムーザは w-Jc スムーザと比較して収束性の点で優位であることが広く知られており，その並列化についても既にいくつかの方法が用いられている．そこで本節では，代表的な二つの既存並列化手法を取り上げ，それらの長所・短所について論じる．なお本節および以降の議論では，辞書式順序付けに基づく標準的かつ逐次的な GS 法や GS スムーザを，他の順序付けなどに基づくものと区別するために逐次 GS 法や逐次 GS スムーザと呼ぶ．

3.2.1 逐次 GS 法と w-Jc 法を併用したハイブリッド (Hybrid) スムーザ

格子空間を部分領域に分割し，それぞれをスレッド（あるいはプロセス）に割り当てる方法により GS スムーザを並列化すると，部分領域の境界部分の格子点にスレッド間のデータ依存関係が生じる．そこで，部分領域の内部の格子点に関しては逐次 GS 法を適用し，領域境界上の格子点については w-Jc 法を用いることにより，スレッド間のデータ依存性を除去する Hybrid スムーザがしばしば用いられている．図 3.1 に Hybrid スムーザによる 4 スレッド並列計算における，逐次 GS 法を用いて計算する格子点 (黒) と，w-Jc 法を用いて計算する格子点 (緑) を示す．この手法は概念が平易で，実装も容易であるが，w-Jc 法の導入による収束性の劣化が問題となる．一般に，並列度（部分領域の数）の増加に伴って，収束性が悪化する傾向がみられ，逐次 GS スムーザと比べて収束に要する反復回数 (V サイクル数) が 2 倍程度にまで大きく悪化する場合がある．

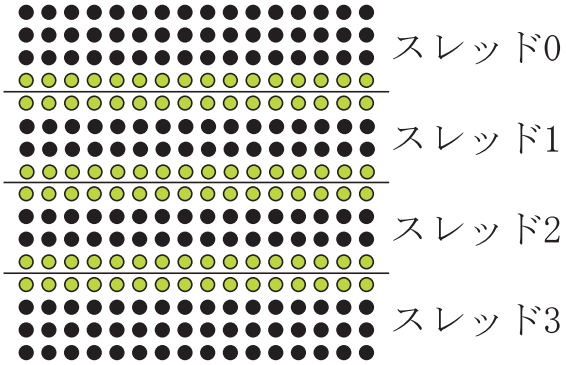


図 3.1: Hybrid スムーザの 1 次元分割並列化

3.2.2 赤-黒順序付けガウスザイデル (RB-GS) スムーザ

3 次元ポアソン方程式の 7 点差分法による離散化によって得られる連立一次方程式では，右辺が $\rho_{i,j,k}$ であるような方程式の左辺に，$\phi_{i,j,k}$ とそれに隣接する格子点の変数のみが出現する．したがって辞書式順序付けに基づく逐次 GS 法の反復計算では，第 2 章の式 (2.17) に示すように，$\phi_{i,j,k}^{(m+1)}$ を求める式の右辺には，格子点 (i,j,k) に隣接する格子点に関する $\boldsymbol{\phi}^{(m)}$ または $\boldsymbol{\phi}^{(m+1)}$ の値のみが出現する．これらの中に $\phi_{i-1.j,k}^{(m+1)}$ などが出現すること，換言すると逐次 GS 法プログラムの配列変数 phi(i,j,k) への代入式の右辺に phi(i-1,j,k) などが出現することにより，w-Jc 法を上回る収束性が得られると同時に，並列化を困難にする依存関係が生じている．

そこで RB-GS 法では，格子点座標 (i, j, k) と近似解ベクトルや右辺ベクトルの要素番号との対応関係，すなわち順序付けを変更することにより，GS 法の簡易な並列化を実現している．具体的には，上記のように $\phi_{i,j,k}^{(m+1)}$ を求める演算で隣接格子点の変数のみが参照されることに着目し，格子点を $(i+j+k) \bmod 2 = 0$ のものと，$(i+j+k) \bmod 2 = 1$ のものに，赤・黒 2 色で塗り分ける．その上で，全ての赤の格子点を辞書式に順序付け，黒の格子点にはそれに続く番号を同じく辞書式順序付けにより与える．このことにより，係数行列 $\boldsymbol{A}$ の狭義下三角行列 $\boldsymbol{L}$ の赤格子点に関する行は必ず 0 となり，また狭義上三角行列 $\boldsymbol{U}$ の黒格子点に関する行は必ず 0 となる．したがって，GS 法の反復式（再掲）

$$\boldsymbol{\phi}^{(m+1)} = \boldsymbol{D}^{-1}\left(\boldsymbol{\rho} + \boldsymbol{U}\boldsymbol{\phi}^{(m)} + \boldsymbol{L}\boldsymbol{\phi}^{(m+1)}\right) \tag{2.16}$$

を，赤と黒の格子点について式 (2.17) と同様に書き下すと，それぞれ以下の

式 (3.1) および (3.2) となる．

$$\phi_{i,j,k}^{(m+1)} = \frac{1}{d}\Big(\rho_{i,j,k} + a\cdot\phi_{i,j,k-1}^{(m)} + b\cdot\phi_{i,j-1,k}^{(m)} + c\cdot\phi_{i-1,j,k}^{(m)} + c\cdot\phi_{i+1,j,k+1}^{(m)} + b\cdot\phi_{i,j+1,k}^{(m)} + a\cdot\phi_{i,j,k+1}^{(m)}\Big) \quad (3.1)$$

$$\phi_{i,j,k}^{(m+1)} = \frac{1}{d}\Big(\rho_{i,j,k} + a\cdot\phi_{i,j,k-1}^{(m+1)} + b\cdot\phi_{i,j-1,k}^{(m+1)} + c\cdot\phi_{i-1,j,k}^{(m+1)} + c\cdot\phi_{i+1,j,k+1}^{(m+1)} + b\cdot\phi_{i,j+1,k}^{(m+1)} + a\cdot\phi_{i,j,k+1}^{(m+1)}\Big) \quad (3.2)$$

上の二つの式は逐次 GS 法の反復式 (2.17) とは異なるが，式 (2.16) の定義に基づいているため GS 法の一種であり，その収束性は標準的なものと同様に良好であると考えられる．実際，均質媒体かつ格子間隔一定の基本的な問題をはじめ多くの実用的な問題に対して，RB-GS 法は逐次 GS 法と同等あるいはそれ以上の収束性を示すことが知られている．

一方，上記の塗り分けにより，$\phi_{i,j,k}^{(m+1)}$ を求める演算では，同色の格子点の値が参照されることはない．したがって，赤の格子点の $\boldsymbol{\phi}^{(m+1)}$ を求める演算は格子点ごとに独立に実施可能であり，黒の格子点についても同様であるため，単純な do-all 型の並列化が可能となる．また，赤・黒それぞれの演算をどのように並列化しても演算結果は不変であるため，RB-GS 法の収束性は並列度には全く依存しない．このため，RB-GS は並列マルチグリッド法のスムーザとして最も一般的に用いられている．

しかし RB-GS 法には，一般的な実装ではメモリ上で不連続となる赤または黒の格子点を連続的にアクセスすることに起因する，キャッシュメモリや主記憶のアクセス機構との不整合という問題点がある．アルゴリズム 1 は，RB-GS 法の標準的な実装方法に基づく Fortran コードであり，最内の i に関するループの増分を 2 とすることによって，赤あるいは黒の格子点の phi(i,j,k) だけを連続的に更新する操作が実現されている．この結果，たとえば phi(i,j,k+1) に関するメモリアクセスは，最内ループの進行に沿って 1 つおきのストライドアクセスとなる．

一方キャッシュメモリは，ラインと呼ばれる一定の大きさの連続アドレスからなるブロックを単位として管理され，主記憶との間でのデータ転送もライン単位で行われる．したがって，一つのラインには赤と黒の格子点に関するデータが同数ずつ共存することとなり，赤あるいは黒の格子点の phi(i,j,k) を更新する三重ループのメモリアクセスのパターンは，ラインを単位に考えると逐次 GS 法の 1 反復分とほぼ同じものになる．この結果，全ての配列データがキャッシュに収まるという極端に小規模の問題を除いて，RB-GS 法の 1 反復あたりのキャッシュミス回数は逐次 GS 法のほぼ 2 倍になる．したがって RB-GS スムーザには，1 回のスムージングに要する時間が逐次 GS スムーザよりも長くなるという問題点がある．

アルゴリズム 1: RB-GS スムーザの一般的な実装例

```
do color = 0, 1
   x_color = color
   y_color = 1 - color
   do k = 2, NZ-1
      do j = 2, NY-1
         do i = 2+x_color, NX-1, 2
            phi(i,j,k)= rho(i,j,k)                          &
          + a(i,j,k)*phi(i,j,k-1) + b(i,j,k)*phi(i,j-1,k) &
          + c(i,j,k)*phi(i-1,j,k) + e(i,j,k)*phi(i+1,j,k) &
          + f(i,j,k)*phi(i,j+1,k) + g(i,j,k)*phi(i,j,k+1) )
          enddo
          x_color = 1 - x_color
      enddo
      x_color = y_color
      y_color = 1 - y_color
   enddo
enddo
```

3.3 ブロック化順序付けによるガウスザイデルスムーザの高速化

3.3.1 ブロック化赤-黒順序付けガウスザイデル (BRB-GS) スムーザ

前節で述べたように，RB-GS スムーザは収束性と並列性の観点では優れているが，ストライドアクセスの悪影響により，ある部分領域に関する 1 回のスムージングに要する時間が逐次 GS スムーザに比べて長くなるという問題点がある．そこで本節では，IC/ILU 分解前処理の並列化手法として提案されたブロック化赤-黒順序付け法 (BRB 順序付け法)[39][44] を，GS スムーザの並列化手法として活用することを提案する．この方法では図 3.2 に示すように，格子空間をまず直方体ブロックに分割し，隣接ブロックが同色とならないように各ブロックを赤・黒の 2 色で塗り分けた上で順序付けを行う．すなわち，まず赤のブロックをブロック単位で辞書式に順序付けし，個々のブロック内部の格子点はブロック内で同じく辞書式に順序付けする．また黒ブロック内の格子点には，赤ブロック内の格子点の中の最大の要素番号に引き続く番号を，同様の順序付けによって付与する．この結果，係数行列 $\boldsymbol{A}$ の下

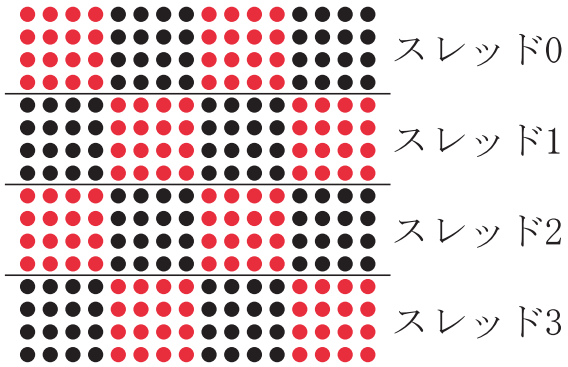


図 3.2: BRB-GS スムーザの 1 次元分割並列化

三角行列 $\boldsymbol{L}$ の赤ブロック格子点に関する行の非ゼロ要素は，同じブロック内の格子点に対応するもののみとなる．同様に上三角行列 $\boldsymbol{U}$ の黒ブロック格子点に関する行の非ゼロ要素も，同じブロック内の格子点に対応するもののみとなる．

したがって，この順序付け法に基づく BRB-GS スムーザは，RB-GS スムーザと同様に一種の GS スムーザであり，また同様の考え方で並列化することができる．すなわち，$\phi_{i,j,k}^{(m+1)}$ を求める演算では，この格子点と同じブロックに属する格子点と，他の色の隣接ブロックに属する格子点のみが参照されるので，図 3.3 に示すようにブロック内の更新演算を逐次 GS スムーザと同様に行いつつ，ブロックを単位とした do-all 型の並列化が容易に実現できる．なお図 3.3 の N_r および N_b は，赤および黒ブロックの個数をそれぞれ意味する．

BRB-GS スムーザの演算結果は，ブロックの形状をある特定のものに固定すれば，ブロック単位の並列化をどのように行っても不変であり， RB-GS スムーザと同様に収束性は並列度には全く依存しない．またブロックサイズを大きくすると，演算過程が定性的には逐次 GS スムーザに近づき，逆にブロックサイズを 1 にすると RB-GS スムーザと一致する．したがって BRG-GS スムーザの収束性は，ブロックの形状・大きさによらず逐次 GS スムーザや RB-GS スムーザと同程度となることが予想される．またこの予想が正しければ，実装に関係する種々のパラメータ，たとえばスレッド数やキャッシュの容量・構成に応じて，適切なブロック形状を選択することもできる．そのようなチューニングは，図 3.3 の N_r と N_b，および逐次 GS スムージングを構成する三重ループの開始・終了インデックスを調整するだけで実現でき，各種 3 次元配列の構造は全く影響されない．

BRB-GS スムーザの RB-GS スムーザに対する最大の優位性は，ブロック内の演算が逐次 GS スムーザと同じであるため，配列データに対するアクセスが連続的になっていることである．したがってブロック形状を適切に定めれば，具体的には x 軸方向のブロックサイズを十分に大きくすれば，BRB-GS

Do $n = 1, N_r$
　n 番目の赤ブロックに対する GS スムージング
End Do
Thread synchronization
Do $n = 1, N_b$
　n 番目の黒ブロックに対する GS スムージング
End Do

図 3.3: BRB-GS の実行手順

スムーザの 1 反復あたりのキャッシュミス回数は，逐次 GS スムーザとほぼ同等となり，1 回のスムージングに要する時間もほぼ同等となる．この事実に，RB-GS スムーザと同等の収束性を示すという予想を加味すると，BRB-GS スムーザは RB-GS スムーザを大幅に上回る性能を示すものと期待できる．

なおブロックの形状には上記のように一定の任意性があるが，マルチグリッド処理との整合性や実装の容易性を勘案し，以下のような（ほぼ）均等なブロック分割に限定している．まず最密格子空間 $\Omega^{(h)}$ の格子点数 NX, NY, NZ は，マルチグリッド処理との整合性から 2 のべき乗に 1 を加えたものとし，$2 \leq i \leq NX-1$, $2 \leq j \leq NY-1$, $2 \leq k \leq NZ-1$ を満たす格子点，すなわち $\partial\Omega$ を除く未知変数が置かれた格子点集合をブロック分割の対象とする．x, y, z の方向に沿ったブロックの個数 n_x, n_y, n_z は，いずれも 2 以上の 2 のべき乗数とし，ブロックサイズは各方向について $nbx = (NX-1)/n_x$, $nby = (NY-1)/n_y$ および $nbz = (NZ-1)/n_z$ とする．ただし未知変数が置かれた格子点の数は $(NX-2) \times (NY-2) \times (NZ-2)$ であるので，全てのブロックが $nbx \times nby \times nbz$ の大きさを持つのではなく，格子点座標が $(2, j, k)$ の格子点を含むブロックについては x 方向のブロックサイズは $nbx - 1$, 同様に $(i, 2, k)$ や $(i, j, 2)$ を含むブロックの y または z 方向のブロックサイズは $nby - 1$ や $nbz - 1$ となる．なお以降の議論では，このような $\partial\Omega$ に隣接する例外的なもの以外のブロックサイズである nbx, nby, nbz を，単にブロックサイズという．またレベル 2 以下の格子空間 $\Omega^{(nh)}$ についても同様のブロック分割を行うが[1]，$\Omega^{(h)}$ のブロック形状と相似なものには限定せず，$\Omega^{(nh)}$ の格子点数やスレッド数などに応じて適切に設定できるものとする．

3.3.2 改良型ブロック化赤-黒順序付けガウスザイデル (mBRB-GS) スムーザ

BRB-GS スムーザの根幹であるブロック分割と BRB 順序付けは，逐次 GS スムーザの特質である配列データの連続アクセスと，RB-GS スムーザの特質

[1] $\Omega^{(nh)}$ の格子点数は $((NX-1)/2^{(n-1)}+1) \times (NY-1)/2^{(n-1)}+1) \times (NZ-1)/2^{(n-1)}+1)$ であるので，$\Omega^{(h)}$ と同様のブロック分割ができる．

Do $n = 1, N_r$
　n 番目の赤ブロックに対する逐次 GS スムージングを α 回行う
End do
Thread synchronization
Do $n = 1, N_b$
　n 番目の黒ブロックに対する逐次 GS スムージングを α 回行う
End do

図 3.4: mBRB-GS の実行手順

である収束性を低下させない並列化とを，同時に達成するために導入したものである．一方，格子空間を部分領域に分割した上で，個々の領域に対して複数回反復のスムージングを局所的に行うシュワルツスムージング [45] を適用すると，マルチグリッド法の V サイクル数の減少に一定の効果があることが知られている．そこで，マルチグリッド法ソルバ全体の性能を向上することを目的として，乗法シュワルツスムージングを導入した改良型の BRB-GS スムーザである mBRB-GS スムーザを提案する．なお乗法シュワルツスムージングは，類似手法である加法シュワルツスムージングが部分領域を完全に独立して扱うのに対して，GS 法が本来規定する部分領域間の依存関係を維持することが特徴であり，BRB 順序付けによるブロック間の依存関係の維持に適合している．

mBRB-GS スムーザは図 3.4 に示すように，BRB-GS スムーザでは 1 反復だけ行われるブロック内の逐次 GS スムージングを，α 回の反復としたものである．なお α 回反復のものを mBRB-GS(α) と表記すると， mBRB-GS(1) スムーザは BRB-GS スムーザと完全に一致する．一方 $\alpha > 1$ の mBRB-GS(α) は，格子空間全体に対して BRB-GS スムージングを α 回反復したものとは異なるスムージング操作を行う．したがって mBRB-GS は厳密な意味では GS 法の一種とは言えないが， V サイクル数の減少効果については BRB-GS や他の GS スムーザの複数回反復と対比した議論ができるものと考えられる．

マルチグリッド法では一般に，1 回のスムージング操作でのスムーザの反復回数 β を増やすことでも，収束までに要する V サイクル数を減少できることが知られている [10]．しかし，多くの場合は格子空間やそれをスレッド単位に分割した部分領域は大きく，そのデータ量はキャッシュ容量を遥かに上回るものとなる．たとえば 65^3 という比較的小さな格子空間または部分領域であっても，1 反復あたり参照される近似解ベクトル，右辺ベクトルおよび各係数のデータ総数は $65^3 \times 9 \approx 2.5 \times 10^6$ 個となり，これが 8 バイトの倍精度浮動小数点データであれば総量は 20 MB 弱となる．この値は，最新のマルチコアプロセッサのコアあたりのキャッシュ容量 2.75 MB を遥かに上回っており，反復計算を cache resident に行うことはできない．したがって各反復の計算時間はほぼ一定となり，スムージングに要する計算時間はほぼ β に比

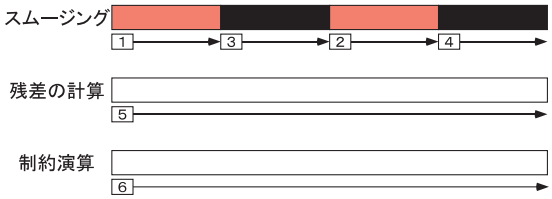
例して増加する．スムージングに要する計算コストはマルチグリッドソルバの計算コストの大部分を占めるため，V サイクル数が $1/\beta$ 以下にならない限り，全体としての性能向上は望めない．しかし現実的には，収束までの V サイクルが $1/\beta$ となるような理想的な効果を得ることは難しいため，ほとんどの場合は反復数を 1 とすることが最適であり，実際にこの設定が用いられるのがほとんどである．

一方 mBRB-GS では，ブロック内の計算に必要なデータ量が各コアが利用可能なキャッシュサイズよりも小さくなるようにブロックサイズを決めることができる．したがって，mBRB-GS スムーザの各ブロック内での第 1 反復の終了時点で，スムージング操作により参照される配列要素が全てキャッシュメモリ上に存在するようにでき，同ブロック内の第 2 反復以降では cache resident な実行となって，第 1 反復に比べて反復あたりの実行時間が大幅に短縮されると期待できる．すなわち，第 1 反復の計算時間を t_s とし，第 2 反復以降の反復あたりの計算時間を $\tilde{t}_s$ としたとき，mBRB-GS(α) に要する計算時間は $t_s + (\alpha - 1)\tilde{t}_s$ となるものと考えられる．また，$\tilde{t}_s \ll t_s$ となることが期待できるため，mBRB-GS(α) スムージングに要する時間は β 回反復の BRB-GS スムージングの計算時間よりも小さいと考えられる．したがって mBRB-GS(α) によって V サイクル数が $1/\alpha$ 以下にはならなくとも，ソルバ全体の速度向上につながる可能性がある．たとえば $t_s = 5\tilde{t}_s$，$\alpha = 3$ とすると，V サイクル数が $(1 + (\alpha - 1)\tilde{t}_s/t_s)^{-1} = 5/7$ 以下になれば，ソルバ全体の実効時間が短縮される．

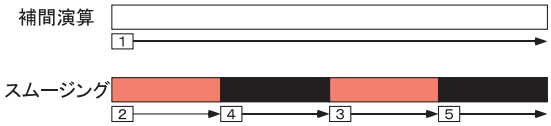
3.3.3 スムーザとその他の演算とのキャッシュブロッキング実装

一般的なマルチグリッドソルバの実装では，スムージング，制約演算，補間演算および残差計算は，それぞれ個別の関数やループにより実現される．このような方法で BRB-GS スムーザを組み込んだマルチグリッドソルバを実装すると，各格子点に関する演算は図 3.5 に示すような形で進行する．なお同図は，1 次元問題の 4 ブロック（赤・黒それぞれ 2 ブロック）による分割を模式的に示しているが，3 次元問題やブロック数が 4 ではない分割の場合にも，本質的に同じ形で計算が進行する．ここである格子点に関する配列要素を含むキャッシュラインに着目すると，格子空間やそれをスレッド単位で分割した部分領域が極端に小さくない限り，たとえばスムージングで参照されたラインが後続の残差計算で再度参照されても，そのラインがキャッシュに残存していることは望めない．またたとえば制約演算では，個々の格子点に関する残差は最大 8 回参照されるが，$NX \times NY$ が大きい値であると，ある $K = \lfloor (k+1)/2 \rfloor$ に関する演算で参照された後に，再び $K + 1$ に関する演算で参照されても，やはりキャッシュに残存していることは望めない．

このような大きな配列の反復的な走査や 3 次元配列を対象とするステンシル計算を，参照局所性を改善して高速化するために，キャッシュブロッキン

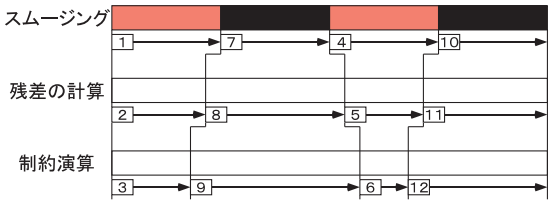


(a) プリスムージングと残差の計算および制約演算

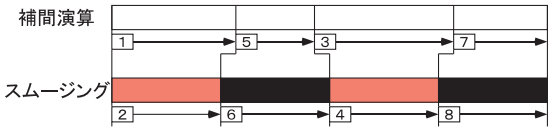


(b) 補間演算とポストスムージング

図 3.5: キャッシュブロッキングを行わない場合の計算順序



(a) プリスムージングと残差の計算および制約演算



(b) 補間演算とポストスムージング

図 3.6: キャッシュブロッキングを行った場合の計算順序

グと呼ばれる手法がしばしば用いられる．この手法では，配列を仮想的に小さなブロックに分割し，ブロックを単位として全体を走査する外側ループと，ブロック内の要素を走査する内側ループにより，配列の全要素走査を実現する．したがって，ブロックの大きさをキャッシュ容量に応じて調整し，ある配列を走査する複数の操作をブロック単位に適用すれば，先行操作での参照によりキャッシュに置かれた配列要素が，後続操作による参照の際にキャッシュに残存することとなる．また一つの操作の中で複数回参照される配列要素についても，2 回目以降の参照がキャッシュにヒットすることが強く期待できる．

このキャッシュブロッキングのマルチグリッドソルバへの適用については，GS スムーザを用いた逐次プログラムの実装方式が提案されており [46]，一定の効果が得られることが示されている．同様の性能改善手法は並列化スムー

Do $n = 1, N_r$
  (i) n 番目の赤ブロックに対する逐次 GS スムージングを α 回行う
  (ii) 部分領域 $(rxs_n+1{:}\, rxe_n-1,\ rys_n+1{:}\, rye_n-1,\ rzs_n+1{:}\, rze_n-1)$ に対する残差計算
  (iii) 部分領域 $(Rxs_n+1{:}\, Rxe_n-1,\ Rys_n+1{:}\, Rye_n-1,\ Rzs_n+1{:}\, Rze_n-1)$ に対する制約演算
End do
Thread synchronization
Do $n = 1, N_b$
  (i) n 番目の黒ブロックに対する逐次 GS スムージングを α 回行う
  (ii) 部分領域 $(bxs_n-1{:}\, bxe_n+1,\ bys_n-1{:}\, bye_n+1,\ bzs_n-1{:}\, bze_n+1)$ に対する残差計算
  (iii) 部分領域 $(Bxs_n-1{:}\, Bxe_n+1,\ Bys_n-1{:}\, Bye_n+1,\ Bzs_n-1{:}\, Bze_n+1)$ に対する制約演算
End do

図 3.7: プリスムージングと残差計算・制約演算のキャッシュブロッキング

ザでも利用可能であり，特に BRB-GS 法と mBRB-GS 法ではスムージングの計算が既にブロック化されているため，キャッシュブロッキングを簡単に導入することができる．

図 3.6 は，BRB-GS スムーザや mBRB-GS スムーザと他の演算に対する，キャッシュブロッキングの適用を概念的に示したものである．プリスムージングでは，あるブロックのスムージングを行った後，引き続いてそのブロックに関する残差計算と制約演算を行い，この一連の操作を全ての赤ブロックと黒ブロックについて繰り返す．またポストスムージングでも同様に，あるブロックに関する補間演算とスムージングを連続して行い，これを全てのブロックについて繰り返す．なおブロック境界付近の格子点を更新対象とする演算については，その演算が参照する格子点が他のブロックに属する場合があるが，後述のように演算の対象ブロックを適切に拡大・縮小することで，順序関係を保った正しいキャッシュブロッキングが実現できる．

図 3.7 は，mBRB-GS スムージングと残差計算・制約演算を対象としたキャッシュブロッキングの，より詳細な手順を示したものである．図中の $(rxs_n{:}\, rxe_n,\ rys_n{:}\, rye_n,\ rzs_n{:}\, rze_n)$ は n 番目の赤ブロックを示す部分配列，また $(bxs_n{:}\, bxe_n,\ bys_n{:}\, bye_n,\ bzs_n{:}\, bze_n)$ は n 番目の黒ブロックを示す部分配列である．また Rxs_n や Rxe_n などは，rxs_n や rxe_n などに対応する 1 レベル下位の格子座標であり，$Rxs_n = \lfloor (rxs_n+1)/2 \rfloor$, $Rxe_n = \lfloor (rxe_n+1)/2 \rfloor$ などと定義される．

図に示すように，まず赤ブロック $(rxs_n{:}\, rxe_n,\ rys_n{:}\, rye_n,\ rzs_n{:}\, rze_n)$ のスムージングを行った後，ブロックを 1 格子点分だけ内側に縮小したブロッ

ク (rxs_n+1: rxe_n-1, rys_n+1: rye_n-1, rzs_n+1: rze_n-1) について残差計算を行う．この縮小により，ブロック境界に接する格子点，すなわち隣接する黒ブロックの格子点の値が残差計算に必要な格子点については，その計算が後に行う黒ブロックの残差計算に委譲される．また続いて行う制約演算も，残差が求まっていない赤ブロックの格子点が関与する部分を黒ブロックの制約演算に委譲するため，1 レベル下のやはり縮小したブロック (Rxs_n+1: Rxe_n-1, Rys_n+1: Rye_n-1, Rzs_n+1: Rze_n-1) を対象として行う．

上記のキャッシュブロッキングされた一連の操作を全ての赤ブロックに対して実施した後，黒ブロックに関する操作を以下のように行う．まず黒ブロック (bxs_n: bxe_n, bys_n: bye_n, bzs_n: bze_n) のスムージングを行った後，ブロックを 1 格子点分だけ外側に拡大したブロック (bxs_n-1: bxe_n+1, bys_n-1: bye_n+1, bzs_n-1: bze_n+1) について残差計算を行う．すなわちこの拡大により，未計算であった赤ブロックの境界格子点に対する計算も実施される．また引き続く制約演算も，未計算であった格子点を含めるために 1 レベル下の拡大したブロック (Bxs_n-1: Bxe_n+1, Bys_n-1: Bye_n+1, Bzs_n-1: Bze_n+1) を対象として実施する．

一方，補間演算とポストスムージングを対象とするキャッシュブロッキングでは，赤ブロックに関する補間演算の際にブロックを拡大し，黒ブロックの演算の際にはブロックを縮小する．すなわち拡大された赤ブロック (rxs_n-1: rxe_n+1, rys_n-1: rye_n+1, rzs_n-1: rze_n+1) に対する補間演算を行うことで，引き続く赤ブロックに対するスムージングで参照される隣接黒ブロック内の境界格子点の値も併せて求める．逆に黒ブロックについては，境界格子点の値は既に計算されているので，スムージングに先立つ補間演算は縮小したブロック (bxs_n+1: bxe_n-1, bys_n+1: bye_n-1, bzs_n+1: bze_n-1) を対象とすればよい．

3.4 スムージングファクタによるスムーザの評価

本節では，既存手法である Hybrid および RB-GS スムーザと，提案手法である BRB-GS および mBRB-GS スムーザを対象に，2.3.3 節で述べた数値実験に基づく近似的なスムージングファクタについて議論する．すなわち対象問題の格子点数を $NX = NY = NZ = 513$ とし，空間 $\Omega^{\{h\}}$ の大きさを $LenX = LenY = LenZ = 1$ として，格子点数が 513 の 1 次元問題を対象とした数値実験により最も減衰しにくいと想定される初期誤差ベクトル $\boldsymbol{o}^{(0)}$ の波数の組 $\tilde{l}_x, \tilde{l}_y, \tilde{l}_z$ を求め，3 次元問題に関する数値実験で求めた $\boldsymbol{o}^{(5)}$ と $\boldsymbol{o}^{(0)}$ のノルム比の 5 乗根をスムージングファクタとして議論する．

まず初めに，Hybrid スムーザのスムージングファクタを求める．Hybrid スムーザを用いたスレッド並列実装では，格子空間を特定の方向（一般には z 軸

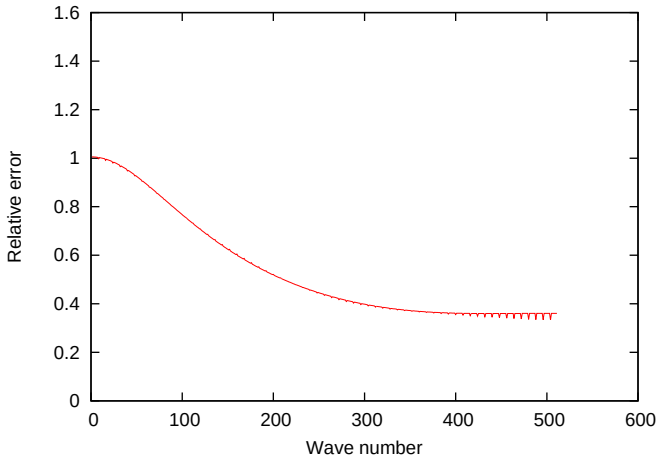


図 3.8: 1 次元問題を対象とした 8 並列 Hybrid スムーザの初期誤差ベクトルの波数と相対誤差の関係

方向）に関してのみ分割する 1 次元分割が一般的であるが，2.3.3 節で議論したようにスムージングが x, y, z のどの方向に対しても対称であるとしなければ，ある波数を持つ誤差ベクトルの減衰を評価することが難しい．そこでここでは，各方向について等しく n 分割する 3 次元分割を対象とし[2]，数値実験は $n = 2, 4, 8$ の各々について行った．したがって並列度（スレッド数）は n^3 であり，それぞれ $2^3 = 8$, $4^3 = 64$, および $8^3 = 512$ となる．またそれぞれに対応する 1 次元問題の並列度は n であり，数値実験により求まった $(\tilde{l}_x, \tilde{l}_y, \tilde{l}_z)$ は，$n = 2$ では $(256, 1, 1)$，また $n = 4$ と $n = 8$ ではともに $(257, 1, 1)$ となった．なお $n = 8$ の実験結果を図 3.8 に示す．またこれらの波数を持つ初期誤差ベクトルを与えた 3 次元問題の数値実験から，$n^3 = 8$ のスムージングファクタは 0.555，また $n^3 = 64$ と $n^3 = 512$ ではそれぞれ 0.566 と 0.585 となり，顕著ではないものの逐次 GS スムーザの 0.542 からの悪化と，並列度を増やすことによる悪化の傾向が認められた．

次に，RB-GS スムーザのスムージングファクタを求める．1 次元問題での波数と相対誤差の関係は図 3.9 に示すものとなり，$(\tilde{l}_x, \tilde{l}_y, \tilde{l}_z)$ は $(342, 1, 1)$ となった．また，この結果を用いた 3 次元問題の数値実験から，スムージングファクタは逐次 GS スムーザよりも小さい 0.498 となった．

BRB-GS スムーザのスムージングファクタはブロックサイズに依存する可能性があるため，$nbx = nby = nbz \in \{2, 8, 32, 128\}$ について値を求める．図 3.10 は，1 次元問題での $nbx \in \{2, 8, 32, 128\}$ のそれぞれについて，波数と相対誤差の関係を示したものである．この図から，nbx が増加すると逐次 GS

[2]実際には 3.3.1 節で述べた BRB-GS スムーザのブロック分割と同様に，$2 \leq i, j, k \leq 512$ を満たす 511^3 個の格子点をほぼ均等な分割の対象とし，たとえば $i = 1$ の yz 境界面に接する部分領域の x 軸方向の大きさは，他の部分領域よりも 1 だけ小さいものとしている．

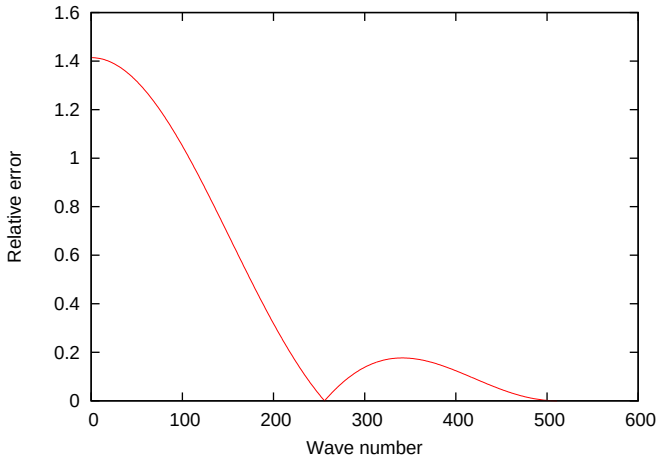


図 3.9: 1 次元問題を対象とした RB-GS スムーザの初期誤差ベクトルの波数と相対誤差の関係

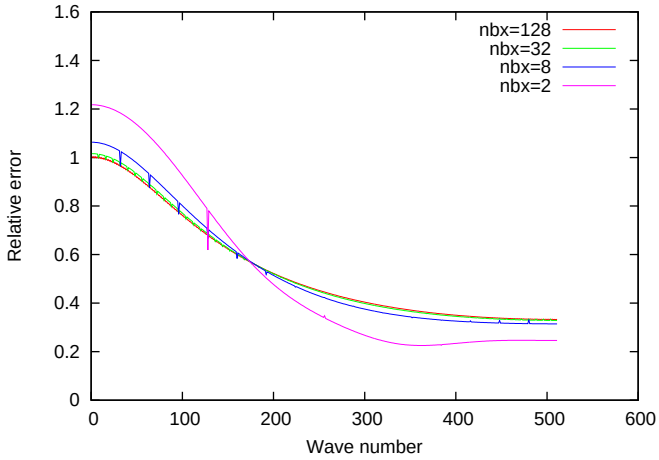


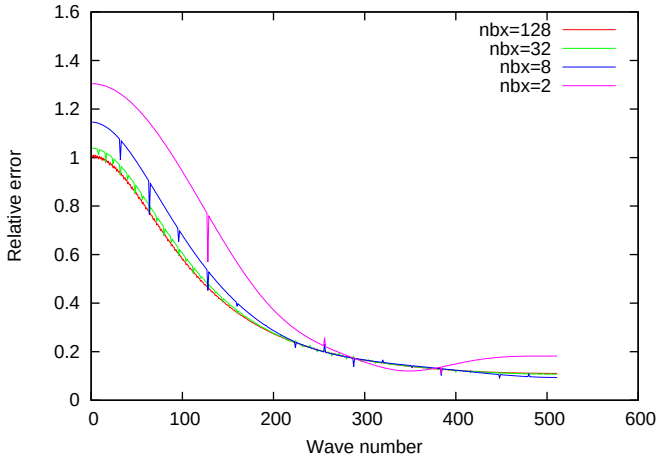
図 3.10: 1 次元問題を対象とした BRB-GS スムーザの初期誤差ベクトルの波数と相対誤差の関係

表 3.1: BRB-GS および mBRB-GS スムーザのスムージングファクタ

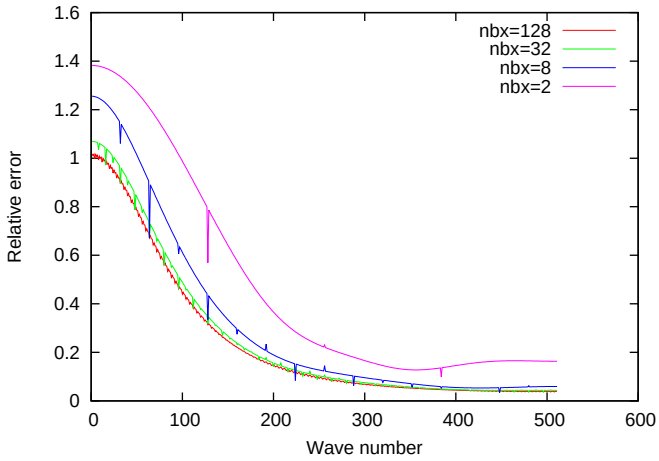
ブロックサイズ	BRB-GS	mBRB-GS	
		α=2	α=3
2^3	0.516	0.353	0.370
8^3	0.540	0.368	0.347
32^3	0.542	0.330	0.308
128^3	0.542	0.307	0.263

スムーザの性質に，また減少すると RB-GS スムーザの性質に，それぞれ近づいて行くことがわかる．具体的には，$l_x \approx 173$ を境に，l_x が小さい範囲ではブロックサイズが大きいときに，逆に l_x が大きい範囲ではブロックサイズが小さいときに，それぞれ相対誤差が小さくなっている．この 1 次元問題に対する数値実験の結果から得た $(\tilde{l}_x, \tilde{l}_y, \tilde{l}_z)$ は，$nbx(= nby = nbz)$ に関わらず $(256, 1, 1)$ であり，この結果から求めたスムージングファクタは，表 3.1 に示すようにブロックサイズを小さくすると小さくなるという傾向を示した．これはブロックサイズを小さくすると，逐次 GS スムーザよりもスムージングファクタが小さい RB-GS スムーザの計算過程に近づくためであると考えられる．

最後に mBRB-GS スムーザのスムージングファクタを求める．図 3.11 は，図 3.10 に示した測定と同じ条件での，波数と相対誤差の関係を $\alpha = 2$ および 3 について示したものである．この図から，ブロックサイズが大きいほど相対誤差が小さく，また α を増やしたときの相対誤差低減効果が大きくなっていることがわかる．また，どのブロックサイズおいても BRB-GS スムーザと同様に $(\tilde{l}_x, \tilde{l}_y, \tilde{l}_z) = (256, 1, 1)$ となり，この結果から求めたスムージングファクタは表 3.1 に示すように，BRB-GS スムーザの場合とは逆にブロックサイズが大きい方が小さくなるという傾向となった．



(a) $\alpha = 2$



(b) $\alpha = 3$

図 3.11: 1 次元問題を対象とした mBRB-GS スムーザの初期誤差ベクトルの波数と相対誤差の関係

表 3.2: HX600 の仕様

ノード仕様	OS	RHEL AS V4
	プロセッサ数	4
	コア数	16
	ピーク演算性能 (DP)	147.2GFlops
	メモリ	DDR2-667 32GB
プロセッサ仕様	シリーズ名称	Quad Core AMD Opteron
	クロック周波数	2.3GHz
	L1 命令キャッシュ	64KB(コアあたり)
	L1 データキャッシュ	64KB(コアあたり)
	L2 キャッシュ	512KB(コアあたり)
	L3 キャッシュ	2MB(共有)

3.5 数値実験結果

本節では，BRB-GS スムーザおよび mBRB-GS スムーザを組み込んでスレッド並列化したマルチグリッド法ポアソンソルバの性能評価実験の結果を，既存のスムーザである Hybrid および RB-GS スムーザによるものとの対比を含めて議論する．

3.5.1 問題設定と実験環境

数値実験に用いたテスト問題は，1 辺の長さが 1 m の立方体状の空間とその中心に設置された半径 3.1cm の球体を対象とするものであり，$\boldsymbol{\rho}$ の値は球体の内部では 1，外部では 0 と設定した．最密の格子空間 $\Omega^{\{h\}}$ の格子数は $NX = NY = NZ = 513$ とし，1 つの V サイクルのレベル数は $L = \log_2(NX - 1) = 9$ とした．初期解ベクトルは $\boldsymbol{\phi}^{(0)} = 0$ とし，収束判定条件は m 回目の V サイクル終了後の残差 $\boldsymbol{r}^{(m)}$ と初期残差 $\boldsymbol{r}^{(0)}$ の一様ノルム比を用いて $\|\boldsymbol{r}^{(m)}\|_\infty / \|\boldsymbol{r}^{(0)}\|_\infty \leq 10^{-7}$ とした．また mBRB-GS スムーザ以外のスムーザについては，プリスムージングとポストスムージングの反復回数 pr と po をともに 1 とした．BRB-GS スムーザを除くスムーザを用いたソルバのスレッド並列化には，格子空間を z 軸に垂直な平面で均等分割する 1 次元分割を用いた．また全てのスムーザについて，レベル 1 から 3 までは最大 16 スレッド，レベル 4 では最大 8 スレッド，またレベル 5 以下では単一スレッドで実行することとした．

それぞれのソルバプログラムは Fortran95 で記述し，スレッド並列化には OpenMP 2.5 を使用した．またコンパイラには Fujitsu Fortran (version 3.2) を使用し，最適化オプションとして O3, Komitfp，Kmfunc，Keval，Kprefetch,

表 3.3: BRB-GS と mBRB-GS の収束性評価実験で用いたレベル 2 以下のブロック数と形状

格子点数	ブロック数			ブロックサイズ		
	x 方向	y 方向	z 方向	x 方向	y 方向	z 方向
257^3 (level 2)	8	8	8	32	32	32
129^3 (level 3)	4	4	4	32	32	32
65^3 (level 4)	2	2	2	32	32	32
33^3 (level 5)	1	1	1	32	32	32
17^3 (level 6)	1	1	1	16	16	16
9^3 (level 7)	1	1	1	8	8	8
5^3 (level 8)	1	1	1	4	4	4

KSSE2，KSSE3，および KOPTERON を与えた．プログラムの実行には，京都大学学術情報メディアセンターのスーパーコンピュータ Fujitsu HX600 の 1 ノードを使用した．このノードは 4 個の 4 コア AMD Opteron プロセッサ (2.3GHz) から構成され，32GB の主記憶を備えている．その他の詳細な仕様は，表 3.2 に示す．また，スレッドのプロセッサコアへの割り当ては，プロセッサあたりのスレッド数が最小となるように，すなわちスレッドが 4 個のプロセッサに分散されるように設定した．

3.5.2 BRB-GS および mBRB-GS スムーザの収束性とブロックサイズの関係

3.3 節で述べたように，収束性の観点からは BRB-GS と mBRB-GS のブロックサイズには一定の任意性があるものと想定され，実行速度の観点で最適なサイズを選択しうると考えられる．このことを確認するために，レベル 1 のブロック形状を立方体とした上で，ブロックサイズ nbx ($= nby = nbz$) を 2, 4, 8, 16, 32, 64, 128 および 256 としたときの，収束までに要する V サイクル数 m を測定した．なお，レベル 2 以下のブロック数は表 3.3 に示す値で固定し，ブロック数が 1 のレベルでは逐次 GS 法を用いた．

測定の結果，BRB-GS については nbx の値に関わらず $m = 11$ となり，mBRB-GS でも同様に nbx に関わらず $\alpha = 2$ では $m = 8$，また $\alpha = 3$ では $m = 7$ となった．したがって，少なくともこの実験に用いたテスト問題と立方体ブロックに関する限り，収束性がブロックサイズに影響されることはないことが確認された．またこのことから，収束性がブロック形状に対しても非依存であると高い確度で推定できるため，次節以降の実験では実行速度の観点で最適と考えられるブロック形状を選択することとした．

表 3.4: BRB-GS の評価に用いたスレッド数とブロック数

スレッド数	x 方向	y 方向	z 方向
1	1	1	2
2	1	2	2
4	1	2	4
8	1	4	4
16	1	4	8

具体的には，まず x 方向のブロック数を 1 に固定することで，格子空間を走査する三重ループの最内のものの回転数を最大化し，連続的にアクセスされるメモリ区間をできるだけ長くするとともに，ループ制御オーバヘッドが最小化されるようにした．またその上で, ブロック形状を定める nby と nbz を，BRB-GS ではスレッド数によって，また mBRB-GS ではキャッシュ容量によって，それぞれ最適と考えられる値に設定することとした

3.5.3 BRB-GS スムーザの性能評価

本節では，BRB-GS スムーザを採用したマルチグリッドソルバの並列性能を，Hybrid および RB-GS スムーザを用いたものと比較しつつ示す．性能評価は，スレッド数を 1, 2, 4, 8, 16 として行った．また BRB-GS のブロック数は表 3.4 に示すように，3.5.2 節の議論に基づき x 方向には 1 とし，y および z 方向については総ブロック数がスレッド数の 2 倍になるように設定した．すなわち，並列度の上限を定める同色ブロック数をスレッド数と一致させることで，ブロックサイズを最大化する設定とした．なお，各スレッドには y 方向に隣接する赤・黒ブロックの対を割り当てたので，4 スレッドまでは z 方向の 1 次元分割，8 および 16 スレッドは y および z 方向の 2 次元分割となっている．

表 3.5 と図 3.12 は，各スムーザを採用したマルチグリッドソルバのスレッド数と性能の関係を，V サイクル数 (表 3.5) と計算時間 (図 3.12) を指標として，それぞれ示したものである．なおスレッド数が 1 の Hybrid スムーザに関する結果は，逐次 GS スムーザによるものを意味している．

Hybrid スムーザを用いたソルバの並列実行では，一部の格子点を w-Jc スムーザにより処理するため V サイクル数が逐次 GS スムーザと比べて約 2 倍に増加している．したがって 2 スレッド実行の計算時間が逐次実行とほとんど変わらず，それ以上のスレッド数での性能向上は比較的良好であるにも関わらず，総体としての並列性能は低くなっている．

RB-GS を用いたソルバでは，スレッド数によらず逐次 GS スムーザと同等以上の収束性を示している．したがって，並列実行での収束性が Hybrid に

表 3.5: スムーザごとのスレッド数と V サイクル数の関係

	スレッド数				
	1	2	4	8	16
Hybrid	11	21	21	21	21
RB-GS	10	10	10	10	10
BRB-GS	11	11	11	11	11

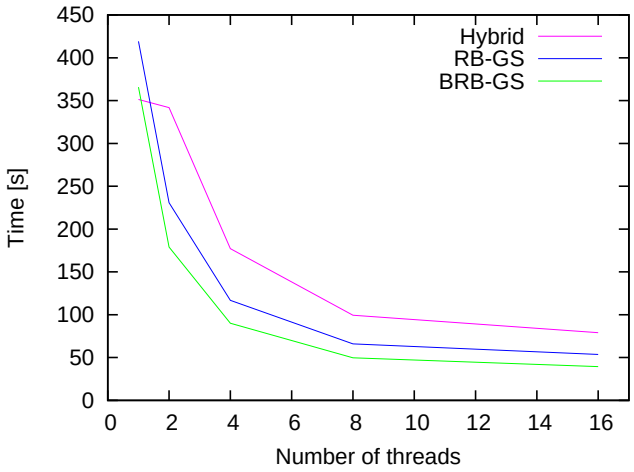


図 3.12: 各スムーザを用いたソルバの計算時間の比較

対して圧倒的に優位であり，それがそのまま計算時間の優位性として表れている．

BRB-GS を用いたソルバでも同様に，逐次 GS 法と同等の収束性が得られており，やはりこのことが Hybrid に対する計算時間の優位性として表れている．一方 RB-GS と比較すると，収束性は同程度であるが，1 反復あたりの計算時間が短縮されるため，かなり大きな性能向上が達成されている．具体的には 16 スレッド実行の性能が，BRB-GS は Hybrid に対して約 2.00 倍，RB-GS に対して 1.36 倍となっている．この結果から，並列実行時に収束性の劣化を招くことなく，かつメモリアクセスの連続性を保つことでスムージング速度も低下しないという，BRB-GS の優れた特性を確認することができた．

3.5.4 mBRB-GS スムーザの性能評価

本節では，mBRB-GS スムーザを採用したマルチグリッドソルバの性能を，ブロックごとのスムージング反復回数 α との関係を中心に議論し，また他のスムーザを用いたソルバとの性能比較を台数効果（逐次・並列性能比）の観

表 3.6: mBRB-GS の評価に用いた各レベルのブロック数と形状

格子点数	ブロック数			ブロックサイズ		
	x 方向	y 方向	z 方向	x 方向	y 方向	z 方向
513^3 (level 1)	1	128	128	512	4	4
257^3 (level 2)	1	32	64	256	8	4
129^3 (level 3)	1	16	16	128	8	8
65^3 (level 4)	1	4	8	64	16	8
33^3 (level 5)	1	2	2	32	16	16
17^3 (level 6)	1	1	1	16	16	16
9^3 (level 7)	1	1	1	8	8	8
5^3 (level 8)	1	1	1	4	4	4

点で行う．3.5.2 節で述べたように，mBRB-GS のブロック形状はキャッシュ容量を基準に定めることができる．実験に用いたプロセッサのコアあたりのキャッシュ容量は，各コア固有である L1 および L2 キャッシュの容量に，4 コアで共有される L3 キャッシュの容量の 1/4 を加えたものと考えられ，表 3.2 に示した数値に基づくと約 1 MB となる[3]．一方，プリスムージング，残差計算および制約演算では，解ベクトル $\boldsymbol{\phi}$，右辺ベクトル $\boldsymbol{\rho}$，行あたりの非ゼロ要素が 7 個である係数行列 $\boldsymbol{A}$，および残差ベクトル $\boldsymbol{r}$ が，それぞれ複数回参照・更新される．そこで，ブロック境界格子点に関する演算でのブロック外の隣接格子点の参照や，3.3.3 節で述べたブロックの拡大・縮小の効果を無視すると，一連の演算により参照・更新される配列要素の数は，ブロックあたりでおよそ $nbx \times nby \times nbz \times 10$ と見積もられ，これに倍精度浮動小数点データのバイト数 8 を乗じた値がキャッシュ容量未満であることが制約条件となる．したがって $nbx \times nby \times nbz \leq 2^{20}/(10 \times 8) \approx 13000$ がおよその制約条件となり，これに nbx, nby, nbz がいずれも 2 のべき乗数である制約を加えると，$nbx \times nby \times nbz = 2^{13} = 8192$ が最大のブロックサイズとなる．

この最大ブロックサイズの見積値と，3.5.2 節で述べた x 方向のブロック数を 1 とする条件に基づいて，V サイクルの各レベルでのブロック数とブロックサイズを定めた結果を表 3.6 に示す．表に示すように，レベル 5 までのブロックサイズは 8192 であり，$ny = nbz$ または $nby = 2nbz$ として両者の比を最小化している．一方レベル 6 以下では，全体の格子点数が 8192 未満となるのでブロック数を 1 としており，そのためスムージングは逐次 GS 法によるものとなっている．

上記のブロックサイズの設定によるキャッシュブロッキングの効果を確かめるために，まず $\alpha = 1$ とした mBRB-GS の性能，すなわち BRB-GS にキャッ

[3]Opteron プロセッサのキャッシュは，各レベルのキャッシュの内容が重複しない排他型であるため，レベルごとの容量の総和を総体としての容量とみなすことができる．

シュブロッキングを適用したものの性能を，16 スレッド実行により測定した．なお本節ではこの mBRB-GS(1) を，$\alpha > 1$ のものと区別するために cBRB-GS と呼ぶ．測定の結果，cBRB-GS の実行時間は 32.49 秒となり，BRB-GS の実行時間 39.47 秒に対して 1.21 倍の性能向上となることから，キャッシュブロッキングの有効性を確認することができた．

次に，mBRB-GS 本来の効果として期待される，スムージング時間の増加を抑制した V サイクル数削減による性能向上について，16 スレッド実行の評価結果に基づき議論する．まず最密格子空間 $\Omega^{(h)}$ を対象として，mBRB-GS(2) の 1 回目と 2 回目のスムージングの計算時間 t_s と $\tilde{t}_s$ を計測した．その結果，$t_s = 0.67$ 秒に対して $\tilde{t}_s = 0.11$ 秒となり，$\tilde{t}_s/t_s \approx 1/6$ という結果が得られた．したがって，$\alpha = 2$ とすることで削減される V サイクル数が 1/2 未満であっても，より具体的には V サイクル数が $(1 + (\alpha - 1)\tilde{t}_s/t_s)^{-1} \times 11 \approx 9.4$ 以下に削減されれば，ソルバ全体の性能が向上することとなる．また α をより大きくすることや，プリスムージングとポストスムージングで異なる α を選択することで，さらに大きな効果が得られることも予想される．

これを確認するためのデータとして，プリスムージングとポストスムージングにそれぞれ mBRB-GS(pr) と mBRB-GS(po) を用いたソルバの計算時間と V サイクル数を，$1 \leq pr, po \leq 10$ の範囲で計測した結果を表 3.7 に示す．この表に示すように，概ね $2 \leq pr, po \leq 6$ の範囲を中心に，cBRB-GS を用いたソルバの実行時間 32.49 秒を有意に短縮する組み合わせが数多く存在する．またこれらのほとんどでは V サイクル数が 6〜8 の範囲であるので，cBRB-GS の V サイクル数 11 からの削減率が 1/2 未満であることから，$\tilde{t}_s/t_s \approx 1/6$ の効果が実行時間の削減に大きく貢献していることも明らかになった．なお，最も良い結果を与えたのは，$(pr, po) = (4, 3)$ の組み合わせであり，V サイクル数は 6，また計算時間は 24.10 秒であった．

次に，逐次 GS スムーザを用いたソルバの計算時間を基準として，各並列化スムーザを用いたソルバの台数効果を図 3.13 に示す．なおこの評価では，mBRB-GS の pr と po の値を，16 スレッド実行については上記の評価に基づいて $(pr, po) = (4, 3)$ とし，また 1〜8 スレッド実行についても同様の評価に基づいて $(pr, po) = (1, 2)$ とした．この設定値の差については後ほど議論する．

この図は，スムーザの違いによる絶対的な性能差を示しているだけでなく，8 スレッド実行と 16 スレッド実行の性能の比がスムーザによって大きく異なることも示している．すなわち，mBRB-GS ではスレッド数が 2 倍になることで性能も 1.78 倍になっているが，cBRB-GS では 1.42 倍，他のスムーザでは 1.23〜1.26 倍という低い値になっている．これは，スムージング，残差計算，制約・補間演算のいずれについても，格子点あたりの計算量に対してメモリからのロード /ストア量が多く，実効演算性能がメモリバンド幅に律速されるためと考えられる．この実験で用いたプロセッサは，コア数が 4 である

表 3.7: mBRB-GS スムーザを用いたソルバの計算時間と V サイクル数 (「計算時間/V サイクル数」と表記)

pr \ *po*	1	2	3	4	5
1	32.40/11	28.18/9	27.00/8	25.61/7	27.55/7
2	28.03/9	26.28/8	24.72/7	26.66/7	24.55/6
3	26.75/8	24.60/7	26.34/7	24.24/6	25.97/6
4	28.85/8	26.45/7	24.10/6	25.79/6	27.54/6
5	31.08/8	28.42/7	25.78/6	27.48/6	29.20/6
6	33.38/8	30.36/7	27.58/6	29.14/6	30.94/6
7	35.74/8	32.42/7	29.29/6	30.98/6	32.71/6
8	38.05/8	34.44/7	31.05/6	32.68/6	28.67/5
9	40.55/8	36.55/7	32.91/6	34.57/6	30.18/5
10	42.95/8	38.84/7	34.72/6	36.26/6	31.75/5

pr \ *po*	6	7	8	9	10
1	29.62/7	31.69/7	33.83/7	35.70/7	37.82/7
2	26.34/6	28.04/6	29.86/6	31.65/6	33.40/6
3	27.66/6	29.47/6	31.14/6	33.09/6	34.81/6
4	29.26/6	30.96/6	27.38/5	28.77/5	30.31/5
5	25.76/5	27.21/5	28.80/5	30.14/5	31.73/5
6	27.18/5	28.71/5	30.22/5	31.65/5	33.13/5
7	28.71/5	30.19/5	31.68/5	33.15/5	34.62/5
8	30.17/5	31.66/5	33.18/5	34.54/5	36.13/5
9	31.71/5	33.07/5	34.58/5	36.09/5	37.54/5
10	33.15/5	34.64/5	36.16/5	37.60/5	39.09/5

のに対しメモリチャネル数は 2 であるため，各コアが物理的なメモリアクセスを頻繁に行うような計算では，2 コアだけを用いる計算と 4 コア全てを用いる計算との間で，大きな性能改善を望むことはできない．したがって，スレッドに割り当てられる配列全体を繰り返し走査する BRB-GS, RB-GS および Hybrid では，8 スレッド（プロセッサあたり 2 スレッド）実行の性能に対して，16 スレッド（プロセッサあたり 4 スレッド）実行の性能の改善度が小さくなっている．また cBRB-GS では，キャッシュブロッキングの効果で物理的なメモリアクセスの頻度が抑制されるため，BRB-GS などに比べると高い改善度となるが，より頻度が低い $(pr, po) = (4, 3)$ の mBRB-GS に比べると改善度が小さくなっている．

一方 mBRB-GS の 8 スレッド性能と 16 スレッド性能の比は，前述のように良好な値となっているが，この事実はスレッド数による pr と po の最適値の違いにも強く相関している．すなわち 8 スレッド以下の実行では，前述のようにメモリバンド幅による律速効果が小さいため，t_s と $\tilde{t}_s$ の比も小さくなる．その結果，pr や po を大きくすることによる V サイクル数削減の効

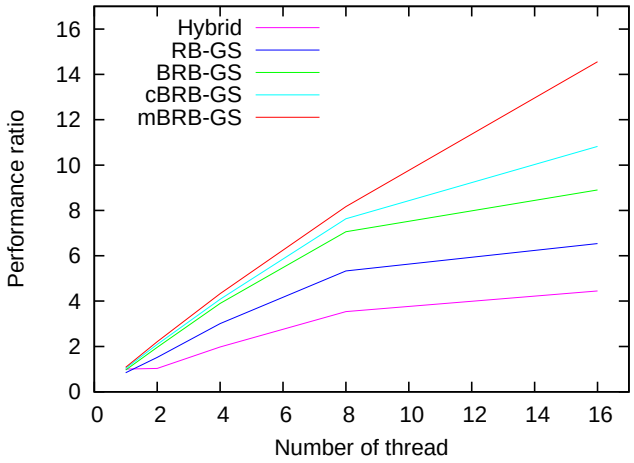


図 3.13: 逐次 GS スムーザによるソルバと比較した各並列化ソルバの台数効果

果が相対的に小さくなり，16 スレッド実行では最適である $(pr, po) = (4, 3)$ が，8 スレッド以下の実行では $(pr, po) = (1, 2)$ よりもやや劣る結果をもたらす．しかし $(pr, po) = (1, 2)$ の設定では，メモリバンド幅の律速効果が大きい 16 スレッド実行での性能改善度が小さく，より大きな改善度が得られる $(pr, po) = (4, 3)$ との間で，表 3.7 に示すような性能の逆転現象が生じている．このように，mBRB-GS 法はコアあたりの実効メモリバンド幅が切迫する状況において有用性が高い手法であると言える．

以上をまとめると，全てのスレッド数に対して mBRB-GS が最も高速であり，以下性能が高い順に cBRB-GS，BRB-GS，RB-GS，Hybrid となることが明らかになった．mBRB-GS の 16 スレッドでの性能は Hybrid に対して 2.88 倍，RB-GS に対して 2.22 倍であり，いずれに対しても大きな優位性が確認できた．また，逐次スムーザと比べると，16 スレッドで 14.6 倍という理想に近い性能向上が得られた．

3.6 まとめ

本章では，3 次元ポアソン方程式の差分解析を対象としたマルチグリッド法の並列化の中心的課題である，スムーザの並列化について議論した．本章に示した重要な貢献の一つは，ブロック化赤-黒順序付け法を用いた並列化 GS スムーザである BRB-GS スムーザの提案と評価である．この BRB-GS スムーザを用いたソルバの 16 スレッド実行では，既存手法である Hybrid スムーザによるものに対して 2.00 倍，また RB-GS スムーザによるものに対して 1.36 倍の性能が得られ，これらの既存手法に対する優位性が確認された．

次に，BRB-GS の改良版として，ブロック内の GS 反復を複数回行う一種の乗法シュワルツスムーザである，mBRB-GS スムーザの提案を行った．mBRB-GS スムーザによるソルバは．BRB-GS スムーザによるソルバよりもさらに高速であり，16 スレッド実行では Hybrid に対して 2.88 倍，RB-GS に対して 2.22 倍の性能が得られた．また mBRB-GS スムーザを用いると，ブロック内の GS 反復数の増加によりメモリアクセスを抑制しながら V サイクル数を削減できる．その結果，コア数の増加がメモリバンド幅拡大をもたらさない計算環境でも並列台数効果を得ることができ，逐次 GS スムーザによるソルバと比較して 16 スレッド実行で 14.6 倍の速度向上を実現した．

また BRB-GS および mBRB-GS スムーザのブロックサイズと，ソルバの収束性（V サイクル数）との関係を数値実験で評価した結果，両者には有意な依存性はないという結論に至った．この結果は，ブロックサイズあるいは形状を，スレッド数やキャッシュ容量などの実装に関するパラメータに応じて最適化できることを意味し，高性能実装の観点からは極めて有益な知見である．ただしこの結果が問題依存である可能性は否定できないため，今後より詳細な解析や多数の実験を行うことで，結論の妥当性の検証あるいはブロックサイズ・形状最適化の新たな指針の検討を実施する．

第4章　クラスタ環境におけるマルチグリッドポアソンソルバの実装と評価

4.1　背景

本章では，前章で示した BRB-GS および mBRB-GS スムーザを用いたマルチグリッド法ソルバのプロセス並列化について議論する．

マルチフィジックスシミュレーションを並列計算環境で実施する動機として，並列処理による計算の加速以外にも，単一のコンピュータでは不可能な問題規模の拡大が挙げられる．一方近年の高性能システムは，1〜2 個という少数のプロセッサからなる共有メモリノードを多数結合した分散メモリ構成が一般的であり，問題規模の拡大に不可欠なメモリ容量の拡大には，複数ノードの使用が不可欠となる．したがって，共有メモリノードの内部だけで実現可能なスレッド並列化では大規模問題に対応することができないため，複数のノードを用いるために分散メモリへの対応，すなわちプロセス並列化が必然となる．また問題規模の拡大は計算量の増大を意味するため，プロセス並列化による計算の加速も当然求められる．

第 1 章でも触れたように，ポアソン方程式の求解法として FFT が用いられることがしばしばあり，プロセス並列化された FFT ライブラリ [47] を用いれば実装も比較的容易である．しかし大規模な 3 次元問題の並列 FFT による求解では，多くの場合は格子空間の 1 次元または 2 次元分割（あるいは FFT 適用のための再分割）と，大域的な 3 次元配列に対する一種の転置操作が必要となる．これらのためのプロセス間通信はいずれも，プロセス全体あるいはプロセス群の中での全体全データ交換通信 (all-to-all communication) となり，プロセスあたりの通信量がプロセスあたりの問題サイズ（プロセスに割り当てられる格子点数）に比例し，かつ遅延が通信に関与するプロセス数に比例することから，大規模な並列計算環境では通信時間がボトルネックとなって並列化効率が低くなることが知られている [8]．

一方マルチグリッド法では，自然で効率的な格子空間の 3 次元分割が容易に実現でき，かつプロセス間の通信は分割された部分領域の境界面に関するデータのみを対象とする．したがって通信量は問題サイズの 2/3 乗のオーダとなり，かつ遅延もプロセス数に関わらず定数オーダとなるため，特に大規

模な並列計算環境に適した求解法であるといえる．

マルチグリッド法ソルバをプロセス並列化するためには，当然スムーザを並列化する必要がある．第 3 章でも述べたように既存の並列化スムーザとして Hybrid スムーザおよび RB-GS スムーザがあり，さらに本稿では BRB-GS スムーザおよび mBRB-GS スムーザを提案している．スレッド並列化したこれらのスムーザの性能が，1 回のスムージングに要する計算時間とマルチグリッド法の収束性 (V サイクル数) の 2 点に大きく影響されることと，両者の観点で優れている mBRB-GS スムーザが最も高速であることを，第 3 章では示した．プロセス並列化ではこれに加えて，格子空間全体のスムージングのために必要なプロセス間の通信量が，並列計算性能に強く影響する．

本章ではまず，Hybrid，RB-GS，BRB-GS，mBRB-GS の各スムーザについて通信量と通信回数を議論し，続いて数値実験によってこれらのスムーザを用いたマルチグリッド法ソルバの性能を評価する．

4.2 マルチグリッド法のプロセス並列化

本節では，プロセス並列化されたマルチグリッド法ソルバが必要とするプロセス間通信について述べる．なお本稿の主な対象はスムーザであるが，スムージング以外の部分でもプロセス間通信が必要である．そこでまず，スムーザの並列化法によらず共通して行われる通信について，同じく共通な領域分割法とともに示す．続いて各スムーザについて，スムージングに必要な通信量と通信回数を議論する．

4.2.1 領域分割とスムージング以外の通信

本章で述べるプロセス並列化では，3 次元の格子空間の各方向それぞれを分割する，3 次元領域分割を対象とする．一般に N^3 の問題空間を P 個のプロセスに分割する場合，1 次元分割では $N \gg P$ が，また 2 次元分割では $N^2 \gg P$ が，それぞれ一定の並列化効率を得るための制約となるのに対し，3 次元分割では $N^3 \gg P$ を満たせばよく，P が大きな大規模並列環境に適している．また一般にプロセス間通信量は分割された部分領域の境界面の面積に比例するが，1 次元分割では $\mathcal{O}(N^2)$，2 次元分割では $\mathcal{O}(N^2/\sqrt{P})$ となるのに対し，3 次元分割では $\mathcal{O}(N^2/P^{2/3})$ であるため最も小さく，並列性能を左右する通信コストの観点でも最適である．

最密格子空間 $\Omega^{\{h\}}$ の大きさを $NX \times NY \times NZ$ としたとき，$P = P_x \times P_y \times P_z$ なる P プロセスに対する格子空間の分割は，以下のように行う．まず，マルチグリッド処理との整合性と実装の簡素化の観点から，$NX-1$, $NY-1$, $NZ-1$ はそれぞれ P_x, P_y, P_z の倍数であるものとし，かつ $n_x = (NX-1)/P_x$, $n_y = (NY-1)/P_y$, $n_z = (NZ-1)/P_z$ はいずれも 2 のべき乗であるとする．

したがって，たとえば NX 個の格子点を P_x 個のプロセスに（ほぼ）均等に分割すると，プロセスあたりの x 方向の格子点数はおよそ n_x となるが，制約・補間演算の並列実装を簡素化するために，プロセスあたりの格子点数を n_x+1 とし，隣接するプロセスに部分領域の境界を重複して割り当てることとしている．すなわち q_x 番目のプロセス $(0 \le q_x < P_x)$ に割り当てられる格子座標の第 1 次元は $q_x n_x + 1 \le i \le (q_x+1)n_x + 1$ の範囲となる．ここで n_x は 2 のべき乗であるので，$\Omega^{\{2h\}}$ については $q_x(n_x/2)+1 \le I \le (q_x+1)(n_x/2)+1$ が，q_x 番目のプロセスに関する第 1 次元の格子座標 I の範囲となる．したがって，両端の格子点については $i = 2I - 1$ の関係，すなわち粗密格子座標の対応関係が満たされる．また $\Omega^{\{3h\}}$ 以下の粗い格子空間についても格子座標の対応関係が保たれ，実際に $L \ge \log_2 \min(n_x, n_y, n_z)$ なる適切なレベル L まで $\Omega^{\{h\}}$ と同様のプロセス分割と部分領域の割り当てを行う．なおこの方法では，求解問題の固定境界 $\partial\Omega$ も含めた格子空間を分割対象としているため，$\partial\Omega$ を含む部分領域を持つプロセスでは，演算対象となる未知変数の数が小さくなる．

図 4.1 は，2 次元の格子空間を対象とする 16 プロセスでの 2 次元分割と，プロセス間通信の様子を模式的に示したものである．前述のように部分領域の境界は，その境界で隣接するプロセスに対して重複して割り当てるため，各プロセスは背景が赤の格子点と，それを囲む背景が緑の境界格子点の計算を担当する．この緑の境界格子点に関するスムージングと残差計算には，隣接する 4 格子点（3 次元問題では 6 格子点）の ϕ の値が必要であるため，それらの中で隣接プロセスが保持するもの（図では背景が青の格子点）をプロセス間通信によって取得する必要がある．また制約演算では隣接する 8 個（3 次元問題では 26 個）の格子点の r を参照するため，部分領域の頂点（3 次元問題では辺と頂点）を共有するプロセスが保持する値（図では背景が黄色）も必要となる．ただし，この頂点を共有するプロセスとの通信を直接行う必要はなく，図に示すように東西方向の通信を行った後に，その通信で得たデータも含めて南北方向の通信を行えば，4 方向（3 次元問題では 6 方向）に隣接するプロセスとの通信だけで実現することができる．したがって，取得すべき値が $\boldsymbol{\phi}$ であるか $\boldsymbol{r}$ であるかに関わらず，3 次元問題では x, y, z の各方向に隣接する 6 プロセスとの通信を行えばよい．

次に，プロセス並列化した V サイクルマルチグリッド法の計算と通信の手順を以下に示す．なお，残差の通信に関係する操作を赤で示し，青で示すスムージングの前後およびその過程で必要な通信は次節以降で述べる．

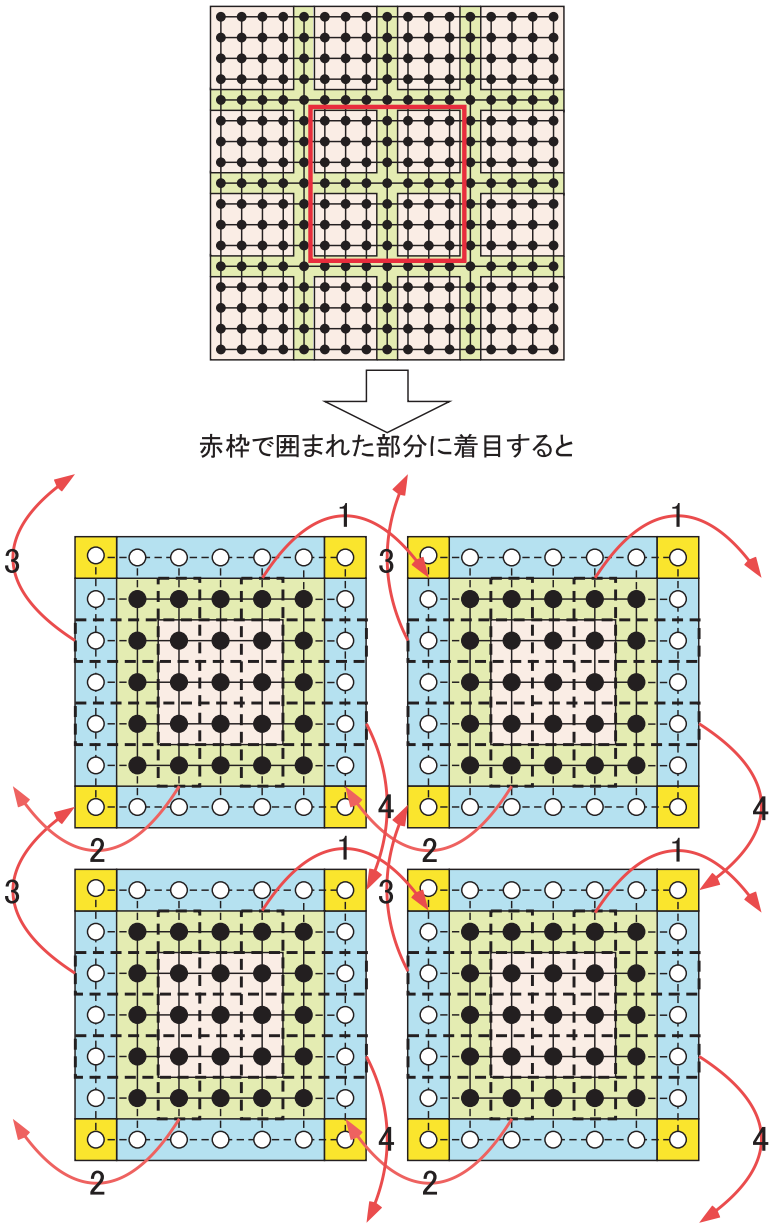


図 4.1: 領域分割とプロセス間通信

$\boldsymbol{A}^{\{h\}}\boldsymbol{\phi}^{\{h\}} = \boldsymbol{\rho}^{\{h\}}$ に基づく反復法を初期解に pr 回適用し $\boldsymbol{\phi}^{\{h\}(pr)}$ を得る

$\boldsymbol{r}^{\{h\}} = \boldsymbol{\rho}^{\{h\}} - \boldsymbol{A}^{\{h\}}\boldsymbol{\phi}^{\{h\}(pr)}$　残差を計算する

隣接プロセス間で境界面内外の $\boldsymbol{r}^{\{h\}}$ を通信

$\boldsymbol{\rho}^{\{2h\}} = \boldsymbol{I}_h^{2h}\boldsymbol{r}^{\{h\}}$　$\Omega^{\{h\}}$ での残差を $\Omega^{\{2h\}}$ に写像する

$\boldsymbol{A}^{\{2h\}}\boldsymbol{\phi}^{\{2h\}} = \boldsymbol{\rho}^{\{2h\}}$ に基づく反復法を初期解 (0 ベクトル) に
pr 回適用し $\boldsymbol{\phi}^{\{2h\}(pr)}$ を得る

$\boldsymbol{r}^{\{2h\}} = \boldsymbol{\rho}^{\{2h\}} - \boldsymbol{A}^{\{2h\}}\boldsymbol{\phi}^{\{2h\}(pr)}$　残差を計算する

隣接プロセス間で境界面内外の $\boldsymbol{r}^{\{2h\}}$ を通信

$\boldsymbol{\rho}^{\{3h\}} = \boldsymbol{I}_{2h}^{3h}\boldsymbol{r}^{\{2h\}}$　$\Omega^{\{2h\}}$ での残差を $\Omega^{\{3h\}}$ に写像する

$\vdots$

$\boldsymbol{\rho}^{\{Lh\}} = \boldsymbol{I}_{(L-1)h}^{Lh}\boldsymbol{r}^{\{(L-1)\}}$　$\Omega^{\{(L-1)h\}}$ での残差を
$\Omega^{\{Lh\}}$ に写像する

$\boldsymbol{\rho}^{\{Lh\}}$ を単一プロセスに集約する

$\boldsymbol{A}^{\{Lh\}}\boldsymbol{\phi}^{\{Lh\}} = \boldsymbol{\rho}^{\{Lh\}}$　最粗格子空間での解を
一つのプロセスが求める

$\boldsymbol{\phi}^{\{Lh\}}$ を全てのプロセスに分配する

$\boldsymbol{o}^{\{(L-1)h\}} = \boldsymbol{I}_{Lh}^{(L-1)h}\boldsymbol{\phi}^{\{Lh\}}$　$\Omega^{\{Lh\}}$ での誤差ベクトルを
$\Omega^{\{(L-1)h\}}$ に写像する

$\vdots$

$\boldsymbol{o}^{\{2h\}} = \boldsymbol{I}_{3h}^{2h}\boldsymbol{\phi}^{\{3h\}(po)}$　$\Omega^{\{3h\}}$ での誤差ベクトルを $\Omega^{\{2h\}}$ に写像する

$\tilde{\boldsymbol{\phi}}^{\{2h\}} = \boldsymbol{\phi}^{\{2h\}(pr)} + \boldsymbol{o}^{\{2h\}}$　誤差ベクトルを用いて $\boldsymbol{\phi}^{\{2h\}(pr)}$ を修正する

$\boldsymbol{A}^{\{2h\}}\boldsymbol{\phi}^{\{2h\}} = \boldsymbol{\rho}^{\{2h\}}$ に基づく反復法を $\tilde{\boldsymbol{\phi}}^{\{2h\}}$ に po 回適用し
$\boldsymbol{\phi}^{\{2h\}(po)}$ を得る

$\boldsymbol{o}^{\{h\}} = \boldsymbol{I}_{2h}^{h}\boldsymbol{\phi}^{\{2h\}(po)}$　$\Omega^{\{2h\}}$ での誤差ベクトルを $\Omega^{\{h\}}$ に写像する

$\tilde{\boldsymbol{\phi}}^{\{h\}} = \boldsymbol{\phi}^{\{h\}(pr)} + \boldsymbol{o}^{\{h\}}$　誤差ベクトルを用いて $\boldsymbol{\phi}^{\{h\}(pr)}$ を修正する

$\boldsymbol{A}^{\{h\}}\boldsymbol{\phi}^{\{h\}} = \boldsymbol{\rho}^{\{h\}}$ に基づく反復法を $\tilde{\boldsymbol{\phi}}^{\{h\}}$ に po 回適用し $\boldsymbol{\phi}^{\{h\}(po)}$ を得る

上記の $\boldsymbol{r}^{\{nh\}}$ のプロセスあたりの通信量（配列要素数）$comm_{\boldsymbol{r}}(n)$ は，前述のようにプロセスの部分領域を各方向に 1 ずつ拡大した領域の表面の格子点数であるので，$n_x^{\{nh\}} = n_x/2^{n-1}$, $n_y^{\{nh\}} = n_y/2^{n-1}$, $n_z^{\{nh\}} = n_z/2^{n-1}$ と

定義すると，以下の値となる．

$$comm_{\boldsymbol{r}}(n) = \left(n_x^{\{nh\}}+3\right)\left(n_y^{\{nh\}}+3\right)\left(n_z^{\{nh\}}+3\right) - \left(n_x^{\{nh\}}+1\right)\left(n_z^{\{nh\}}+1\right)\left(n_z^{\{nh\}}+1\right) \quad (4.1)$$

後述する $\boldsymbol{\phi}^{\{nh\}}$ の通信も含め，$n < L$ なるレベル n では隣接プロセスの間で部分領域境界面の内外に隣接する格子点の値を交換する局所的な通信のみが行われるが，レベル L では全ての大域的な通信が必要となる．すなわち各プロセスが保持する $\boldsymbol{r}^{\{Lh\}}$ をある特定のルートプロセスに集約する通信 (gather) と，ルートプロセスが求めた $\boldsymbol{\phi}^{\{Lh\}}$ を全プロセスに分配する通信 (scatter) が必要となる．一般にこれらの通信は $\mathcal{O}(\log P)$ のステップ数を要し，かつルートプロセスでは $\mathcal{O}(P)$ の大きさのデータを送受信することになる．したがって，各プロセスが担当する部分領域の大きさを一定に保ちながらプロセス数を拡大する形で問題の大規模化を図った場合，ある時点でレベル L での大域的な通信コストの影響が問題となると予想される．しかし現時点で一般的な 1000 プロセス程度までの並列計算では，L を適切に設定することにより，ルートプロセスの送受信量をレベル 1 での通信量と比較して十分小さな値に抑えることができる．たとえば 4.3 節で述べる最大規模の実験では，$n_x = n_y = n_z = 256$, $P_x = P_y = P_z = 6$ $(P = 216)$, $L = \log_2 n_x = 8$ という設定であるので，ルートプロセスが収集する $\boldsymbol{r}^{\{Lh\}}$ の大きさはプロセスあたりで $(n_x^{\{Lh\}})^3 = (256/2^7)^3 = 2^3 = 8$ となり[1]，総計では $8 \times 216 = 1728$ となる．この値は，たとえば $\boldsymbol{r}^{\{h\}}$ のプロセスあたりの通信量 $comm_{\boldsymbol{r}}(1) = (256+3)^3 - (256+1)^3 = 399386$ の 1/200 以下であり，V サイクル 1 回に要する通信全体に対して無視できる量である．

4.2.2 Hybrid スムーザ

Hybrid スムーザによる $\Omega^{\{nh\}}$ でのプリスムージングの計算・通信の手順は，以下に示すものとなる．

隣接プロセス間で境界面内外の $\boldsymbol{\phi}^{\{nh\}}$ を通信 ($n = 1$ のみ)
部分領域の境界面の $\boldsymbol{\phi}^{\{nh\}}$ に対して w-Jc スムージングを行い，
　結果を一時配列に格納する
内部領域の $\boldsymbol{\phi}^{\{nh\}}$ に対して逐次 GS スムージングを行う
一時配列に保存しておいた値を用いて境界面の $\boldsymbol{\phi}^{\{nh\}}$ を更新する
隣接プロセス間で境界面内外の $\boldsymbol{\phi}^{\{nh\}}$ を通信

[1]隣接プロセスと共有している境界面のデータ収集は不要であるため $(n_x^{\{Lh\}}+1)^3$ ではなく $(n_x^{\{Lh\}})^3$ となる．

上記の $\boldsymbol{\phi}^{\{nh\}}$ に関する 1 回目の通信は $n = 1$ の場合にのみ行われ，直前の V サイクルで得られた $\boldsymbol{\phi}^{\{h\}}$ の部分領域外の値を，w-Jc スムージングで参照するために隣接プロセスから取得する[2]．また 2 回目は n に関わらず行われ，スムージング後の残差計算で参照する値を隣接するプロセスから取得する．したがって 1 回の通信についてプロセスあたりの通信量（配列要素数）$comm_{\boldsymbol{\phi}}(n)$ は，たとえば x 軸に垂直な境界面の格子点数が $(n_y^{\{nh\}}+1)(n_z^{\{nh\}}+1)$ となることから，$\partial\Omega$ を部分領域内に含まないプロセスについて，以下の値となる．

$$comm_{\boldsymbol{\phi}}(n) = 2\left\{\left(n_x^{\{nh\}}+1\right)\left(n_y^{\{nh\}}+1\right)+\left(n_x^{\{nh\}}+1\right)\left(n_z^{\{nh\}}+1\right) + \left(n_y^{\{nh\}}+1\right)\left(n_z^{\{nh\}}+1\right)\right\} \quad (4.2)$$

一方ポストスムージングでは，部分領域外の格子点に関する $\tilde{\boldsymbol{\phi}}^{\{nh\}}$ の値を取得するために，1 回目の通信が n に関わらず行われる．しかし引き続く補間演算では，部分領域内の $\boldsymbol{\phi}^{\{nh\}(po)}$ のみの参照で部分領域内の $\boldsymbol{o}^{\{(n-1)h\}}$ が計算できるため，2 回目の通信は不要である．なおこの通信を省略できることが，隣接プロセス間で部分領域境界面を共有する理由の一つである．

以上をまとめると，Hybrid スムーザの $\Omega^{\{nh\}}$ に関する通信回数 $\gamma(n)$ は 3 $(n = 1)$ または 2 $(n > 1)$，また通信量の総計は $\gamma(n) \cdot comm_{\boldsymbol{\phi}}(n)$ となる．

4.2.3 RB-GS スムーザ

RB-GS スムーザによる $\Omega^{\{nh\}}$ でのプリスムージングの計算・通信の手順は，赤・黒それぞれの格子点の $\boldsymbol{\phi}^{\{nh\}}$ を $\boldsymbol{\phi}^{r\{nh\}}$ および $\boldsymbol{\phi}^{b\{nh\}}$ と表記すると，以下に示すものとなる．

隣接プロセス間で境界面内外の $\boldsymbol{\phi}^{b\{nh\}}$ を通信 $(n = 1$ のみ$)$
$\boldsymbol{\phi}^{r\{nh\}}$ の更新を行う
隣接プロセス間で境界面内外の $\boldsymbol{\phi}^{r\{nh\}}$ を通信
$\boldsymbol{\phi}^{b\{nh\}}$ の更新を行う
隣接プロセス間で境界面内外の $\boldsymbol{\phi}^{b\{nh\}}$ を通信

Hybrid スムーザと同様に，上記の 1 回目の通信は $n = 1$ の場合にのみ行われるが，赤の格子点 $\boldsymbol{\phi}^{r\{h\}}$ の更新で参照されるのは黒の格子点 $\boldsymbol{\phi}^{b\{h\}}$ のみであるので，通信対象も $\boldsymbol{\phi}^{b\{h\}}$ に限定される．2 回目の通信は赤の格子点 $\boldsymbol{\phi}^{r\{h\}}$ の更新結果を，黒の格子点 $\boldsymbol{\phi}^{b\{h\}}$ の更新で参照するためのものであるため，n に関わらず行われる．また 3 回目の通信は Hybrid スムーザと同様に，残差の計算で参照する値を取得するためのものであるが，赤の格子点 $\boldsymbol{\phi}^{r\{h\}}$ につ

[2] $n > 1$ のプリスムージングでは，初期解ベクトルを常に 0 とするため，通信は不要である．

いては 2 回目の通信で取得するため，3 回目の通信は黒の格子点 $\boldsymbol{\phi}^{b\{h\}}$ に対してのみ行われる．

ポストスムージングでは，Hybrid スムーザと同様に 1 回目の通信が n に関わらず行われ，また 2 回目の通信も常に行われるが，3 回目の通信はやはり不要である．したがって，通信回数 $\gamma(n)$ は 5 $(n=1)$ または 4 $(n>1)$, また $\boldsymbol{\phi}^{r\{h\}}$ と $\boldsymbol{\phi}^{b\{h\}}$ の通信量がともに $comm_{\boldsymbol{\phi}}(n)/2$ であることに留意すると[3]，また通信量の総計は $\gamma(n)\cdot comm_{\boldsymbol{\phi}}(n)/2$ となる．すなわち，通信回数は Hybrid スムーザよりも増加するが，通信量は同じ $(n>1)$ または 5/6 $(n=1)$ となる

4.2.4 BRB-GS および mBRB-GS スムーザ

BRB-GS スムーザの $\Omega^{\{nh\}}$ でのプロセス間通信は，基本的には RB-GS スムーザと同様に，赤・黒それぞれのブロック内の格子点に関する $\boldsymbol{\phi}^{\{nh\}}$ である $\boldsymbol{\phi}^{rb\{nh\}}$ と $\boldsymbol{\phi}^{bb\{nh\}}$ について，その更新に必要な部分領域外の $\boldsymbol{\phi}^{\{nh\}}$ を取得するものである．しかし図 4.2 に示すように，あるプロセスの部分領域境界のうち格子点座標が小さい側（下部境界）ではブロック境界と部分領域境界が一致するのに対し，座標が大きい側（上部境界）では主に隣接プロセスが保持するブロックの境界部分の格子点が部分領域に含まれるため，下部境界と上部境界では通信対象のブロックの色が異なる．すなわち図に示すように，ある色のブロックのスムージングで参照される部分領域外の格子点は，下部境界では他色のブロックに属するのに対して，上部境界では同色のブロックに属する[4]．これを勘案し，下部境界面に隣接する部分領域外の赤・黒ブロック格子点の $\boldsymbol{\phi}^{\{nh\}}$ をそれぞれ $\boldsymbol{\phi}^{rb\{nh\}-}$ と $\boldsymbol{\phi}^{bb\{nh\}-}$ とし，また上部境界面について同様に $\boldsymbol{\phi}^{rb\{nh\}+}$ と $\boldsymbol{\phi}^{bb\{nh\}+}$ とすると，プリスムージングの計算・通信の手順は以下に示すものとなる．

$\boldsymbol{\phi}^{bb\{nh\}-}$, $\boldsymbol{\phi}^{bb\{nh\}+}$, $\boldsymbol{\phi}^{rb\{nh\}+}$ を隣接プロセスから取得（$n=1$ のみ）

$\boldsymbol{\phi}^{rb\{nh\}}$ の更新を行う

$\boldsymbol{\phi}^{rb\{nh\}-}$ を隣接プロセスから取得

$\boldsymbol{\phi}^{bb\{nh\}}$ の更新を行う

$\boldsymbol{\phi}^{bb\{nh\}-}$, $\boldsymbol{\phi}^{bb\{nh\}+}$, $\boldsymbol{\phi}^{rb\{nh\}+}$ を隣接プロセスから取得

なお RB-GS スムーザと同様に，1 回目の通信は $n=1$ のプリスムージングと全てのポストスムージングで，2 回目の通信は全てのプリスムージングとポストスムージングで，また 3 回目の通信はプリスムージングでのみ，それぞれ行われる．したがって，通信回数 $\gamma(n)$ は 5 $(n=1)$ または 4 $(n>1)$, ま

[3] 境界面の格子点数は奇数であるため，厳密には赤の格子点について $(comm_{\boldsymbol{\phi}}(n)-1)/2$, 黒の格子点について $(comm_{\boldsymbol{\phi}}(n)+1)/2$ となる．

[4] 厳密には，ある方向のブロックサイズが 1 である場合，その方向に垂直な上部境界面についても他色のブロックの格子点が参照される．

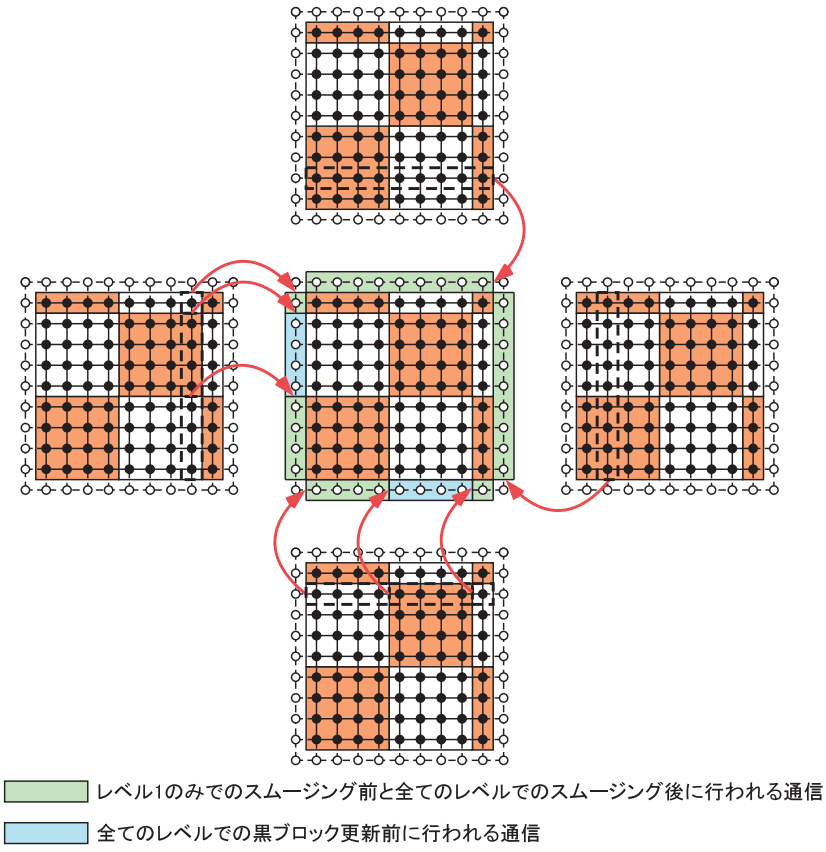


図 4.2: BRB-GS のプロセス並列化での必要な隣接通信

た $\phi^{rb\{h\}\pm}$ とまた $\phi^{bb\{h\}\pm}$ の通信量がいずれもほぼ $comm_{\phi}(n)/4$ であることを勘案すると，$n = 1$ では $(11/4) \cdot comm_{\phi}(1)$，$n > 1$ では $2 \cdot comm_{\phi}(n)$ となる．すなわち，Hybrid スムーザと比較すると通信回数は増加するが，通信量は同じ $(n > 1)$ または 1/12 だけ小さく $(n = 1)$ なり，RB-GS スムーザと比較すると通信回数は同じで通信量も同じ $(n > 1)$ または 1/10 だけ大きく $(n = 1)$ なる．

一方，mBRB-GS スムーザによる $\Omega^{\{nh\}}$ でのプリスムージングの計算・通信の手順は，以下に示すものとなる．

$\phi^{bb\{nh\}-}$, $\phi^{bb\{nh\}+}$, $\phi^{rb\{nh\}+}$ を隣接プロセスから取得（$n = 1$ のみ）

$\phi^{rb\{nh\}}$ の更新を α 回行う

$\phi^{rb\{nh\}-}$ を隣接プロセスから取得

$\phi^{bb\{nh\}}$ の更新を α 回行う

$\phi^{bb\{nh\}-}$, $\phi^{bb\{nh\}+}$, $\phi^{rb\{nh\}+}$ を隣接プロセスから取得

上記のように，α 回のブロック内 GS スムージングの間で隣接する他色ブロックの $\boldsymbol{\phi}^{\{nh\}}$ は変化しないため，隣接プロセスとの通信は α に依存しない．したがって $\alpha = 1$ である BRB-GS スムーザと全く同じ通信が行われ，通信回数・通信量も BRB-GS スムーザと完全に一致する．

以上，4.2.2 節から本節までの議論をまとめると，通信回数の観点では Hybrid スムーザが最小であり，他の 3 スムーザは同等かつ Hybrid の 2 倍程度となり，通信量の観点では RB-GS が最小，BRB-GS と mBRB-GS がそれに次ぎ，Hybrid が最大となる．実際の通信時間はこの二つのファクタに通信対象データのメモリ上の配置，たとえば RB-GS はあらゆる境界面でデータが不連続であるのに対して他のスムーザでは 4 境界面について一定長の連続データとなることや，ノード内外の通信を司るハードウェアやソフトウェアの機構にも依存するため，通信に関するスムーザ間の優劣を論じることは難しい．ただし，たとえば $n_x = n_y = n_z = 256$ という設定では $comm_{\boldsymbol{\phi}}(1) \approx 3\,\mathrm{MB}$ と大きな値であって，10 GB/s の高速通信路を介しても 300 μs 程度を要することと，通信回数に比例するファクタである 1 回あたりの遅延が数 μs であることを勘案すると，通信回数の差は大きな影響力を持たないものと考えられる．また通信量についても，4.2.1 節で述べた残差 $\boldsymbol{r}^{\{nh\}}$ の通信量が全てのスムーザで同等であることを考えると，$n = 1$ で生じるスムーザ間の差は 10 % 未満となり，大きなものではない．したがって通信時間の観点では 4 つのスムーザはほぼ同等と考えられ，第 3 章で議論したスレッド並列化と同様に，1 回のスムージングに要する計算時間と V サイクルの数が性能を支配するものと予想される．

4.3 数値実験結果

本節では，BRB-GS スムーザおよび mBRB-GS スムーザを組み込んでプロセス並列化したマルチグリッド法ポアソンソルバの性能評価実験の結果を，既存のスムーザである Hybrid スムーザおよび RB-GS スムーザによるものとの対比を含めて議論する．

4.3.1 問題設定と実験環境

数値実験に用いたテスト問題は，1 辺の長さが 1m の立方体状の空間とその中心に設置された半径 7.8mm の球体を対象とするものであり，$\boldsymbol{\rho}$ の値は球体の内部では 1，外部では 0 と設定した．最密の格子空間 $\Omega^{\{h\}}$ の格子数は，プロセスあたりの数を $n_x = n_y = n_z = 256$ と固定し，各方向のプロセス数も $P_x = P_y = P_z$ とした上で，$1 \le P_x \le 6$ の範囲でプロセス数を変化させる weak scaling により定めた．すなわち $NX = NY = NZ = 256P_x + 1$ であり，格子点数の増加に応じて格子間隔が短縮する設定となっている． 1 つの V サ

表 4.1: GreenBlade8000 の仕様

システム仕様	ノード数	601
	ピーク演算性能	242.5TFlops
	総メモリ容量	38TB
	二分バンド幅	3.1TB/s
ノード仕様	OS	RHEL V6.2
	プロセッサ数	2
	コア数	16
	メモリ	DDR3-1600 64GB
	ノード間結合網	InfiniBand FDR x 2
プロセッサ仕様	シリーズ名称	Intel Xeon E5
	クロック周波数	2.6GHz
	L1 命令キャッシュ	32KB(コアあたり)
	L1 データキャッシュ	32KB(コアあたり)
	L2 キャッシュ	256KB(コアあたり)
	L3 キャッシュ	20MB(共有)

イクルのレベル数は $L = \log_2 n_x = 8$，初期解ベクトルは $\boldsymbol{\phi}^{(0)} = 0$ とし，収束判定条件は m 回目の V サイクル終了後の残差 $\boldsymbol{r}^{(m)}$ と初期残差 $\boldsymbol{r}^{(0)}$ のユークリッドノルム比を用いて $\|\boldsymbol{r}^{(m)}\|_2/\|\boldsymbol{r}^{(0)}\|_2 \leq 10^{-7}$ とした．また mBRB-GS スムーザ以外のスムーザについては，プリスムージングとポストスムージングの反復回数は $(pr, po) = (1, 1)$ とした．

それぞれのソルバプログラムは Fortran2003 で記述し，プロセス並列化には MPI 2.0 準拠の Intel MPI (version 4.0) を使用した．またコンパイラには Intel Fortran Composer XE (version 12.1) を使用し，最適化オプションとして-O3 -xHost -no-opt-prefetch を与えた．プログラムの実行には，京都大学学術情報メディアセンターのスーパーコンピュータ GreenBlade 8000 を最大 14 ノード使用した．このシステムは表 4.1 に示すように，2 個の Intel Xeon E5 プロセッサ (Sandy Bridge) を有する総計 16 コアの共有メモリノードを単位として構成されている．またノード間の接続のために，ノードあたり 2 本の InfiniBand FDR リンクが装備されており，ノードの片方向通信速度は 13.56 GB/s である．なお，1 つの MPI プロセスを 1 個のコアに割り当てる flat MPI の実行形式を採用した．

4.3.2 実験結果

表 4.2 は，Hybid, RB-GS および BRB-GS スムーザを採用したマルチグリッドソルバのプロセス数と性能の関係を，計算時間と V サイクル数（カッコ内

表 4.2: スムーザごとのプロセス数と計算時間および V サイクル数の関係

	プロセス数					
	1	8	27	64	125	216
Hybrid	8.02(10)	31.39(17)	24.06(13)	32.73(17)	30.34(15)	38.20(17)
RB-GS	6.97 (8)	23.37 (9)	23.52 (9)	26.46(10)	28.10(10)	31.07(10)
BRB-GS	6.30 (9)	17.87(10)	19.90(11)	20.12(11)	22.14(11)	24.52(11)

表 4.3: BRB-GS および mBRB-GS の性能評価に用いた各レベルのブロックサイズ

	ブロックサイズ					
	BRB-GS			mBRB-GS		
プロセスあたりの格子点数	x 方向	y 方向	z 方向	x 方向	y 方向	z 方向
257^3 (level 1)	256	128	128	256	4	2
129^3 (level 2)	128	64	64	128	4	4
65^3 (level 3)	64	32	32	64	8	4
33^3 (level 4)	32	16	16	32	8	8
17^3 (level 5)	16	8	8	16	8	8
9^3 (level 6)	8	4	4	8	4	4
5^3 (level 7)	4	2	2	4	2	2

の数値）を指標として示したものである．なお前述のように weak scaling による評価であるので，プロセス数に比例して問題規模が増加する．したがって，プロセス数によらず一定の計算時間が得られることが理想的な結果を意味する．またプロセス数が 1 の Hybrid スムーザに関する結果は，逐次 GS スムーザによるものを意味している．さらに，BRB-GS のブロックサイズは表 4.3 に示すように，全てのレベルで x, y, z 方向のブロック数がそれぞれ 1,2,2 となるように設定した[5]．

Hybrid スムーザでは，逐次実行では反復回数が 10 回であるのに対し，並列実行での収束性の悪化（V サイクル数の増加）が顕著である．その結果，スレッド並列の結果と同じように，他の手法よりも計算時間が長くなっている．次に RB-GS スムーザを用いたソルバでは，プロセス数によらず V サイクル数が他のスムーザによるものよりも小さく，収束性の観点では最良の結果となっている．しかし，Hybrid に比べて 1 反復あたりの計算時間が長く，たとえば 64 プロセス実行では Hybrid が 1.85 秒であるのに対し，RB-GS は

[5] 上部境界面のブロックを除外したブロック数である．

表 4.4: mBRB-GS の pr, po と計算時間および反復回数の関係
(1 プロセス実行)

pr \ po	1	2	3	4	5
1	6.50/9	7.04/8	7.28/7	8.39/7	9.50/7
2	7.05/8	7.28/7	7.20/6	8.15/6	9.10/6
3	7.29/7	7.19/6	8.15/6	9.09/6	8.38/5
4	8.40/7	8.13/6	9.10/6	8.38/5	9.17/5
5	9.50/7	9.09/6	10.04/6	9.17/5	9.96/5

表 4.5: mBRB-GS の pr, po と計算時間および反復回数の関係
(64 プロセス実行)

pr \ po	1	2	3	4	5
1	21.35/11	17.70/9	16.50/8	17.65/8	16.54/7
2	17.71/9	15.98/8	14.62/7	15.66/7	16.78/7
3	18.60/9	14.65/7	15.27/7	13.99/6	14.90/6
4	17.66/8	15.65/7	14.04/6	14.90/6	15.76/6
5	18.88/8	16.67/7	14.88/6	15.78/6	16.67/6

約 1.4 倍の 2.65 秒を要している

一方 BRB-GS スムーザを用いたソルバでは，1 反復あたりの計算時間が Hybrid スムーザによるものと同程度（64 プロセスでは 1.83 秒）であるとともに，RB-GS によるものと同程度の収束性が得られている．この高い収束性とスムージング性能による BRB-GS スムーザの優位性は，スレッド並列化の結果とも整合するものである．この結果，全てのプロセス数について BRB-GS スムーザの Hybrid スムーザと RB-GS スムーザに対する優位性が確認され，216 プロセスでは Hybrid よりも 1.56 倍，RB-GS よりも 1.27 倍，それぞれ高速であるという結果が得られた．

mBRB-GS スムーザについては，まず 3.5.4 節と同様の考え方により表 4.1 に示した L2 キャッシュの容量 256 KB を基準として，レベル 4 までのブロックサイズを $nbx \times nby \times nbz = 2048$ となるように定めた．なお表 4.1 に示したように，実験に用いたプロセッサは 8 コアで共有される 20 MB の L3 キャッシュを持っており，コアあたりの容量 2.5 MB を基準とすれば $nbx \times nby \times nbz = 16384$ となる．しかし L3 キャッシュを利用することは，アクセス遅延が L2 キャッシュの 2 倍以上であることからスムージング性能の面で必ずしも有効ではなく，また 3.5.2 節で議論したようにブロックサイズを大きくすることでソルバの収束性が向上することも見込めないため，L2 キャッシュの容量を基準としてブロックサイズを定めた．

次に，プリスムージングとポストスムージングの反復回数 pr と po の性能

表 4.6: mBRB-GS スムーザを用いたソルバの最適な pr と po の値と計算時間および V サイクル数の関係

	プロセス数					
	1	8	27	64	125	216
(pr, po)	$(1,1)$	$(2,2)$	$(2,2)$	$(3,4)$	$(4,3)$	$(3,4)$
反復回数	9	7	7	6	6	6
計算時間	6.50	13.61	13.81	13.99	14.26	14.69

に対する影響を，1 プロセスおよび 64 プロセス実行について $1 \leq pr, po \leq 5$ の範囲で調査した．表 4.4 に示す 1 プロセス実行の結果では，$(pr, po) = (1,1)$ が最も高性能となっている．これは 50 GB/s 以上の大きなメモリバンド幅を有するプロセッサで，1 コア・1 プロセスだけが計算を行っている状況下では，メモリバンド幅に大きな余裕がありキャッシュへのプリフェッチも有効に作用することから，第 1 反復と第 2 反復以降の計算時間に大きな差が生じないことに起因する．実際，第 1 反復の計算時間 t_s と第 2 反復以降の反復あたりの計算時間 $\tilde{t}_s$ の間には，有意な差を確認することができなかった．

一方 64 プロセス実行では表 4.5 に示すように，$(pr, po) = (3,4)$ が最も良い性能を与える設定となり，$(pr, po) = (1,1)$ に比べると 1.53 倍の性能が得られた．これは，全てのコアにプロセスが割り当てられるために生じるメモリバンド幅の逼迫が，$t_s/\tilde{t}_s = 7.4$ という大きな落差をもたらし，かつ V サイクル数削減によりプロセス間通信時間も比例して削減されることによるものである．このようにプロセス数が大きい場合には $pr, po > 1$ とする効果が現れ，表 4.6 に示すように 8 プロセス以上では，BRB-GS スムーザによるものよりも 1.31〜1.67 倍の高い性能が得られた．

以上の結果をまとめ，各スムーザによるソルバの性能を，Hybrid スムーザによるソルバの 1 プロセス実行を基準とした weak scaling の台数効果，すなわちスムーザ S によるソルバの P プロセスでの実行時間を $T(P,S)$ としたときの $P \cdot T(1, \mathrm{Hybrid})/T(P,S)$ の値を，図 4.3 に示す．mBRB-GS スムーザによるソルバは，どのプロセス数の場合でも他の全てのスムーザによるものよりも良好な性能を示しており，216 プロセスの場合では Hybrid の 2.60 倍，RB-GS の 2.12 倍，BRB-GS の 1.67 倍の性能が得られた．

4.4 まとめ

本章ではまず，マルチグリッド法ポアソンソルバのプロセス並列化のための領域分割法と，Hybrid, RB-GS, BRB-GS および mBRB-GS の各並列スムーザに共通および固有のプロセス間通信について議論し，通信回数と通信量の観点からは通信時間に大きな差はないという見通しを得た．続いて各スムー

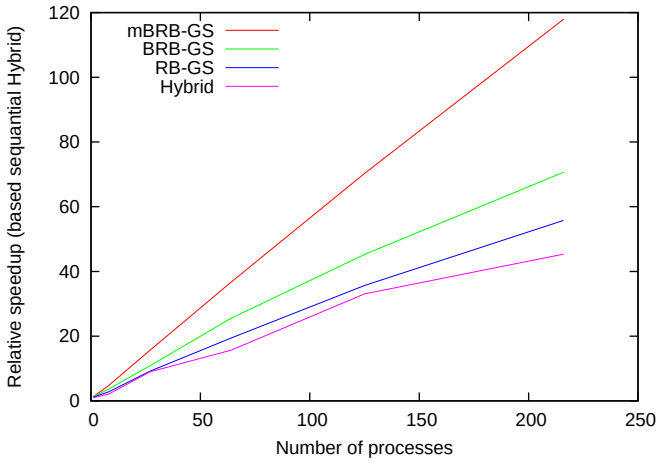


図 4.3: 逐次 GS スムーザによるソルバを基準とした各並列化ソルバの台数効果

ザを用いたソルバのプロセス並列実行の性能について，以下の評価結果を示した．まず BRB-GS をスムーザを用いたソルバは，Hybrid スムーザによるものに対して収束性で，RB-GS スムーザによるものに対して 1 反復あたりの計算時間で，それぞれ優位であることが確認され，216 プロセス実行では Hybrid スムーザによるものよりも 1.56 倍，RB-GS スムーザによるものよりも 1.27 倍の，高い性能であるという結果を得た．また，mBRB-GS スムーザを用いたソルバでは，プロセス数が 8 以上の場合に 2 回以上のプリスムージングとポストスムージングが有効に作用し，BRB-GS スムーザによるものよりもさらに高い性能が得られることを示した．具体的には，216 プロセス実行の性能は，Hybrid の 2.60 倍，RB-GS の 2.12 倍，BRB-GS の 1.67 倍という結果が得られ，第 3 章で議論したスレッド並列化と同様に mBRB-GS スムーザのプロセス並列化での優位性が確認できた．

第5章　メニーコアプロセッサによる高性能なマルチグリッドポアソンソルバ

5.1　背景

第3章では，マルチグリッド法の並列スムーザとしてBRB-GSとmBRB-GSの提案を行い，既存手法であるHybridやRB-GSに対する優位性を，スレッド並列環境とプロセス並列環境について第3章と第4章でそれぞれ示した．本章では，Intel社が近年発表し注目されているXeon Phiプロセッサを対象として，提案手法の有用性の検証と既存手法との比較を行う．

Xeon Phiの倍精度浮動小数点演算の理論性能は1TFlops以上であり，この値は同世代の汎用的なプロセッサと比べて数倍の高い値となっている．また，同程度の性能を持つアクセラレータであるGPGPUでは専用のコーディングを行う必要があるのに対して，Xeon Phiは64ビットx86系の命令セットであるx86_64に類似した命令セットを採用しているため，一般のプロセッサとほとんど同じコードを実行できるというメリットも持っている．これに加えて電力あたりの性能も高く，独立型のXeon Phiの開発も行われていることなどから，今後のHPC分野において重要な役割を果たすと考えられる．

Xeon Phiの高い性能は，60コア240スレッドという高い並列度と，512ビットという幅の広いSIMD演算によって実現されている．SIMDとはSingle Instruction Multiple Dataの略であり，1つのSIMD命令によって複数のデータに対して同時に演算を施すことができる．またXeon Phiは一般的なプロセッサと同じく，階層キャッシュやTLBなどからなる階層型のメモリ構造を持っている．したがって，Xeon Phiで高い性能が得られるプログラムは，240スレッドまでスケールする高い並列性，SIMD演算に適合したループ構造，およびキャッシュを有効活用する高いメモリ参照局所性の，全てを兼ね備えている必要がある．

本稿で提案しているmBRB-GSスムーザを用いたマルチグリッド法ソルバが，高い並列性とメモリ参照局所性を兼ね備えていることは，前章までの議論で明らかにしてきた．しかしmBRB-GSスムーザの核であるブロック内の逐次GSスムージングのループ構造は，SIMD演算には適合しないという問題点がある．SIMD演算は並列演算の一種であるので，ループの各反復の間

表 5.1: Xeon Phi 7120 の仕様

プロセッサ	コア数	60
	クロック周波数	1.24GHz
	L1 データキャッシュ	32KByte(コアあたり)
	L2 キャッシュ	512KByte(コアあたり)
メモリ	技術規格	GDDR5
	容量	16GByte

のフロー依存，すなわちループ運搬フロー依存が存在すると，SIMD 演算を適用することが不可能あるいは著しく困難である．したがって，近似解ベクトルの参照・更新に依存関係を持つ逐次 GS スムージングを自然な形で実装すると，ループ運搬フロー依存が生じるため SIMD 演算を活用することができない．Xeon Phi の実効演算性能と理論演算性能の比の上限は，実際に 1 命令で実行される倍精度浮動小数点演算の数とその最大値である 8 との比であり，SIMD 演算を全く活用できない場合には高々理論演算性能の 1/8 しか得られないことから，実装上の工夫により活用の方策を見出すことが重要である．

そこで本章では，部分的に SIMD 演算が適用可能な GS 法の実装と，この実装の mBRB-GS スムーザへの適用を議論する．この部分的な SIMD 演算の適用，すなわち部分的 SIMD 並列化は，GS 法の自然な実装である空間三重ループの最内ループに対して 6 個の加算を対象とするループ分割 [48] を行い，得られる 6 個のループの内の 5 個に対して SIMD 並列化を行うものである．以下本章では，まず Xeon Phi のアーキテクチャについて SIMD 演算機構を中心に述べ，続いて上記の部分的 SIMD 並列化について，既存手法である重み付きヤコビ（以後 w-JC と表記）および RB-GS スムーザの SIMD 並列化と対比しつつ議論する．またこれら三つの SIMD 並列化したスムーザを用いたマルチグリッド法ソルバの性能を比較評価し，Xeon Phi についても mBRB-GS が優位であることを示す．

5.2 メニーコアコプロセッサ Xeon Phi

表 5.1 に Xeon Phi 7120 の仕様を，図 5.1 にコア内部のアーキテクチャの概略図を，それぞれ示す．Xeon Phi7120 はプロセッサ毎に 60 のコアを有しており，各コアはリングバスと呼ばれる通信路で接続されている．Xeon Phi 7120 の倍精度浮動小数点演算性能の理論ピーク値は 1.19TFlops であり，同世代の汎用プロセッサ Xeon E5 2670 の 7 倍以上の値となっている．本節では，Xeon Phi で高い実効性能を得るために必要と考えられる点について述べる．

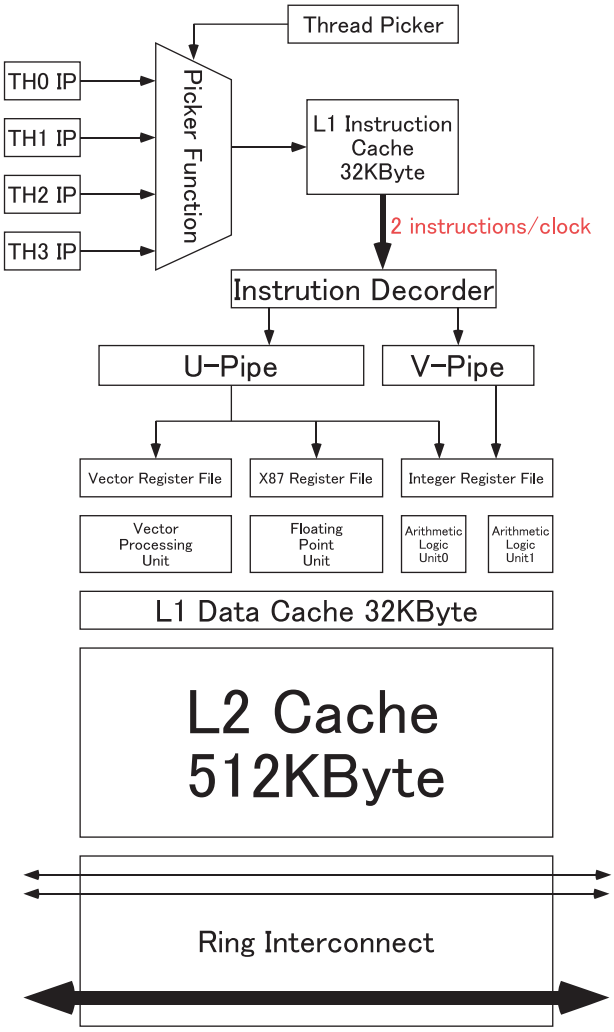
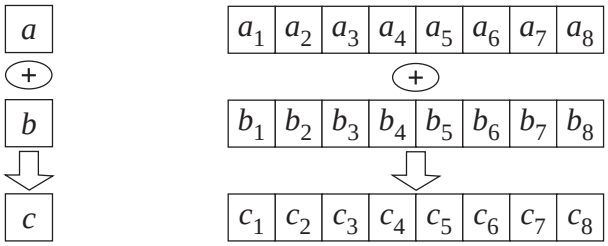


図 5.1: Xeon Phi のコア・アーキテクチャ概略図



(a) スカラー加算　　(b) SIMD 加算

図 5.2: スカラー加算と SIMD 加算

Xeon Phi には一つのコアが複数のスレッドを並行実行するための Hyper-Threading の機構が備えられており，コアあたり最大 4 つのスレッドを実行することができる．Thread Picker はラウンドロビン方式でこれらのスレッドの内の一つを選択し，この選択されたスレッドの命令を Picker Function (PF) がクロックあたり最大 2 個発行する．ただし，PF は一つのスレッドの命令を 2 クロック連続して発行することができないため，クロックごとに命令を発行するためには最低 2 スレッドの並行実行が必要となる．

また Xeon Phi の命令パイプラインはインオーダ制御であり，PF による命令発行順序は命令セットアーキテクチャが規定する順序（プログラム順）と完全に一致する．そのため，先行命令の演算遅延などのために後続命令にデータを供給できないデータハザードが生じた場合，後続命令の発行が停止するだけではなく，それに引き続く全ての命令も発行されない．汎用的なマイクロプロセッサでは，データハザードが生じた命令に続く命令の中に実行可能なものがあれば順序を入れ替えて発行するアウトオブオーダ制御によりハザードに対処するが，Xeon Phi では Hyper-Threading によってハザードによる実行スループットの低下を防ぐ．すなわち複数のスレッドの命令をラウンドロビン方式で発行することにより，同一スレッドの命令実行間隔を拡大し，ハザードの発生頻度自体を抑える方式となっている．したがって命令の実行スループットを高く保つためには，多くの場合は各コアで実行するスレッド数を最大値の 4 とする必要があり，プロセッサ全体では 240 スレッドの並列実行が必要となる．

Xeon Phi の最大の特徴は，512 ビットという幅の広い SIMD 演算機構を持っていることである．SIMD 演算は一種の並列演算であり，一つの命令によって複数のデータに対して同一の演算操作を施すことを意味する．図 5.2 は，基本的な命令による（スカラー）加算 (a) と，Xeon Phi を含む最近のプロセッサが持つ SIMD 加算 (b) を対比して示したものである．図に示すように，スカラー加算は単一オペランドの組 a と b の和を，やはり単一のオペランド c に格納するのに対し，SIMD 加算では複数（図では 8 個）の a_i と b_i の組の和 $a_i + b_i$ を対応するオペランド c_i に格納する．何個の演算を 1 命令で同

時に実行できるかは，機種により異なる最大ビット幅と各オペランドのビット幅で定まるが，Xeon Phi の最大ビット幅は現在の x86 系マイクロプロセッサの中では最も広い 512 ビットであり，8 個の 64 ビットの倍精度浮動小数点数に対する加減乗算を 1 命令でかつ毎サイクル実行することができる．なお，ロード・ストアやメモリオペランドに対する演算も．8 個の倍精度浮動小数点数に対して行うことができるが，8 個のオペランドはメモリ空間上で連続的に配置されていなければならない．

また Xeon Phi は毎サイクル実行可能な積和演算命令も有しており，1 サイクルあたり 8 個の乗算と 8 個の加算を同時に処理することができる．したがって，Xeon Phi 7120 のクロック周波数 1.24 GHz, 倍精度浮動小数点数に関する SIMD 演算幅 8，積和演算による乗算・加算の重複実行度 2，およびコア数 60 を乗じることにより，倍精度浮動小数点演算性能のピーク値が $1.24 \times 10^9 \times 8 \times 2 \times 60 = 1.19$ TFlops と算出される．逆に SIMD 演算が有効活用できず，全ての命令が単一のオペランドの組に対するものである場合，実効的な演算性能はピーク値の高々1/8 となる．このことから，Xeon Phi の高いピーク演算性能を有効に活用するためには，SIMD 並列化が極めて重要であるといえる．

一方 Xeon Phi のメモリ階層構造は，一般的なマイクロプロセッサと同様であり，表 5.1 に示したように各コアは 32 KB の L1 データキャッシュと 512 KB の L2 キャッシュを備え，また全コアで共有される 16 GB のメモリ[1]が装備されている．このメモリは，一般のマイクロプロセッサのものとは異なる GDDR5 規格によるものであり，44 GB/s の転送性能を持つ 8 本のチャネルによりプロセッサと接続されているため，理論最大バンド幅は 352 GB/s である．この値は，たとえば同世代の Xeon E5 2670 の 51.2 GB/s と比較すると約 7 倍であるが，ピーク演算性能の比が 7 倍以上であることを考えると，演算性能との相対値としては同等あるいはやや劣ることとなる．したがって一般のマルチコアプロセッサと同様に，キャッシュを有効に活用して物理的なメモリアクセス頻度をできるだけ抑えることが重要である．

5.3 スムーザの SIMD 並列化

3.2 節でも述べたように，マルチグリッド法の 1 レベルの処理を構成するスムージング，残差計算，制約演算および補間演算の中で，スムージング以外の操作は do-all 型の並列化が容易であり，したがって SIMD 並列化を阻害する本質的な要因はない．しかしスムージングについては，逐次 GS スムーザのようにループ運搬フロー依存を伴うため SIMD 並列化が困難な場合もある．また第 3 章や第 4 章で議論したように，1 回のスムージングに要する時間やマルチグリッド法ソルバ全体の収束性は，どのスムーザを用いるかによって

[1] メモリ容量は機種により異なり，この値は Xeon Phi 7120 のものである．

大きな幅があり，単に SIMD 並列化の難易度だけでスムーザの良否を判断することはできない．そこで本節ではまず，SIMD 並列化が容易な既存スムーザである w-Jc および RB-GS スムーザを取り上げ，これらを採用することの得失について議論する．続いて本稿で提案している mBRB-GS スムーザについて，ブロック内の逐次 GS スムージングを部分的に SIMD 並列化する実装方法を提案する．

5.3.1 既存スムーザの SIMD 並列化

3.2 節で触れたように，w-Jc スムーザでは近似解ベクトルの更新操作に要素間の依存関係が存在しない．具体的には w-Jc 法の反復式（再掲）

$$\boldsymbol{\phi}^{(m+1)} = \omega \boldsymbol{D}^{-1} \left\{ \boldsymbol{\rho} + (\boldsymbol{U} + \boldsymbol{L}) \right\} \boldsymbol{\phi}^{(m)} + (1-\omega)\boldsymbol{\phi}^{(m)} \tag{2.32}$$

の右辺には $\boldsymbol{\phi}^{(m+1)}$ が出現せず，多くのまた自然な実装では $\boldsymbol{\phi}^{(m)}$ と $\boldsymbol{\phi}^{(m+1)}$ は異なる配列により表現される．したがって，$\boldsymbol{\phi}^{(m+1)}$ を表現する配列要素の更新順序は完全に任意となり，メモリ空間上で連続する複数要素の更新に必要な操作を一連の SIMD 演算により容易に実現することができる．一方，第 2 章で定量的評価も含めて議論したように，w-Jc 法の収束性が GS 法よりも劣ることはよく知られており，この点がマルチグリッド法ソルバの性能に悪影響を及ぼす可能性が高い．

一方 3.2.2 節で述べたように，RB-GS スムーザも do-all 型の並列化が容易であり，また第 3 章および第 4 章で示したように，マルチグリッド法ソルバの収束性の観点では非常に優れている．しかし RB-GS スムーザには，3.2.2 節で述べたストライドアクセスの問題に加え，SIMD 並列化に関してもそれに類似した性能上の問題が存在する．すなわち，RB-GS スムーザが連続的に更新する同色の近似解ベクトル要素はメモリ空間上で一つおきに配置され，また更新操作の中での係数行列や他の近似解ベクトル要素の参照も一つおきとなる．したがって，たとえば赤格子点の要素を更新するループでも，赤・黒が混在した 8 個のデータをロードして全てに演算を施し，マスク操作によって本来必要な黒格子点に基づく演算結果だけを赤格子点の要素に書き込むという方法を採用すると，処理スループットは w-Jc の 1/2 となってしまう．あるいは赤・黒混在の 16 個のデータをロードし，必要な黒格子点のデータ 8 個を抽出して詰め合わせる gather 操作を行い，それらに演算を施した後に最終的な結果を一つおきに広げる scatter 操作を行った後で，マスク操作により赤格子点の要素だけに書き込む方法もある．しかしこの方法も，12〜13 個[2]の配列要素に対する gather が必要となる．したがっていずれの方法を用いても，

[2]ある赤格子点 (i, j, k) を先頭とする一連の更新操作で，phi(i-1,j,k) と phi(i+1,j,k) を先頭とする一連の黒格子点の値が重複していることを利用できれば 12 個，できなければ 13 個である．

アルゴリズム 2: 逐次 GS スムージングの一般的な実装

```
do k = zs, ze
  do j = ys, ye
    do i = xs, xe !This loop is not SIMDized
      phi(i,j,k)= rho(i,j,k)                                          &
             + a(i,j,k)*phi(i,j,k-1) + b(i,j,k)*phi(i,j-1,k) &
             + c(i,j,k)*phi(i-1,j,k) + e(i,j,k)*phi(i+1,j,k) &
             + f(i,j,k)*phi(i,j+1,k) + g(i,j,k)*phi(i,j,k+1) )
    enddo
  enddo
enddo
```

スムージング 1 回あたりの時間は w-Jc スムーザよりも大幅に長くなると考えれる．

5.3.2 mBRB-GS スムーザの SIMD 並列化

第 3 章で述べたように mBRB-GS スムーザは，収束性，スレッド並列化への適合性，および参照局所性の観点では優れた手法である．しかし各ブロックに対して逐次 GS スムージングを行うため，そのフロー依存性が SIMD 並列化の阻害要因となる．そこで本節では，逐次 GS スムージングに対して一部を除いて SIMD 並列化が可能となる実装手法を提案する．

アルゴリズム 2 は，ブロックに対する逐次 GS スムージングを Fortran で記述した，一般的な実装例である．ここで，phi と rho はそれぞれ近似解ベクトルと右辺ベクトルに相当する配列， a，b，c，d，e，f，g は係数行列を保持する配列である．また xs, ys, zs はブロックの始点を，xe, ys, ze はブロックの終点を，それぞれ保持する変数である．このコードの最内ループに着目すると，c(i,j,k)*phi(i-1,j,k) の項がループ運搬フロー依存性を持っているため，SIMD 並列化を行うことができない．しかし逆に言えば，最内ループで SIMD 並列化を阻害しているのはこの加算項だけである．そこでアルゴリズム 3 に示すように，最内ループに対して加算項ごとのループ分割 [48] を適用すると，赤で示した c(i,j,k)*phi(i-1,j,k) に関するループを除いて SIMD 並列化が可能となる．

ただしループ分割のためには，部分和を保持するための一時的配列変数 tmp が必要になり，アルゴリズム 2 には存在しない配列の更新・参照操作が最内ループの反復あたり合計で 10 回行われるという問題が新たに生じる．しかし，一時配列 tmp の大きさは x 方向のブロックサイズに等しく，またブロック全体の大きさは各コアが利用可能なキャッシュサイズより小さくなるように設定

アルゴリズム 3: 部分的な SIMD 並列化可能な逐次 GS スムージングの実装

```
do k = zs, ze
  do j = ys, ye
    !$DEC SIMD
    do i = xs, xe
      tmp(i) = rho(i,j,k) + a(i,j,k)*phi(i,j,k-1)
    enddo
    !$DEC SIMD
    do i = xs, xe
      tmp(i) = tmp(i) + b(i,j,k)*phi(i,j-1,k)
    enddo
    !$DEC SIMD
    do i = xs, xe
      tmp(i) = tmp(i) + e(i,j,k)*phi(i+1,j,k)
    enddo
    !$DEC SIMD
    do i = xs, xe
      tmp(i) = tmp(i) + f(i,j,k)*phi(i,j+1,k)
    enddo
    !$DEC SIMD
    do i = xs, xe
      tmp(i) = tmp(i) + g(i,j,k)*phi(i,j,k+1)
    enddo

    do i = xs, xe !This loop is not SIMDized
      phi(i,j,k) = tmp(i) + c(i,j,k)*phi(i-1,j,k)
    enddo
  enddo
enddo
```

されるため，結果として tmp の大きさは L1 キャッシュ容量に比べて十分に小さくすることができる．実際 5.4 節の実験では，x 方向に最長である $\Omega^{\{h\}}$ のブロックサイズは $512 \times 2 \times 2$ であるので，tmp の大きさ $512 \times 8 \approx 4\,\mathrm{KB}$ は L1 キャッシュ容量 32 KB よりも十分に小さい．また tmp は全てのループで参照されるため，8-way のセット連想度を持つ Xeon Phi の L1 キャッシュから追い出されることはない．したがって，アクセス遅延が最小である L1 キャッシュに対する更新・参照操作の増加は，性能に対して悪影響をほとんど及ぼさないものと期待できる．

なおループ分割は，元来ベクトルプロセッサを対象としたコード最適化手法であり，演算性能との相対値が非常に大きなメモリアクセスバンド幅を前提として，反復あたりの計算が複雑であるためベクトル化が困難なループを，走査対象の配列が巨大であっても単純に分割するものであった．しかしこのような単純な分割では，前述のように演算性能に対する相対的なメモリバンド幅が高くない Xeon Phi では，一時配列の導入に起因する物理的なメモリアクセスの増加によって SIMD 並列化の効果が打ち消される，あるいはかえって性能劣化を招く可能性が高い．したがって上記のように，mBRB-GS スムーザの特徴であるブロック化を生かして L1 キャッシュを活用するループ分割は，Xeon Phi や今後のメニーコアプロセッサに適した新しい最適化手法であるとも言える．

表 5.2: mBRB-GS の評価に用いた各レベルのブロック数と形状

格子点数	ブロック数			ブロックサイズ		
	x 方向	y 方向	z 方向	x 方向	y 方向	z 方向
513^3 (level 1)	1	256	256	512	2	2
257^3 (level 2)	1	64	128	256	4	2
129^3 (level 3)	1	32	32	128	4	4
65^3 (level 4)	1	8	16	64	4	8
33^3 (level 5)	1	4	4	32	8	8
17^3 (level 6)	1	1	2	16	16	8
9^3 (level 7)	1	1	1	8	8	8
5^3 (level 8)	1	1	1	4	4	4

5.4 数値実験結果

本節では，5.3.2 節で示した部分的に SIMD 並列化可能な mBRB-GS スムーザの実装である mBRB-Alg2 を用いたマルチグリッド法ソルバの性能を，一般的な逐次 GS スムージングの実装方法に基づく mBRB-Alg1，および 5.3.1 節で示した二つの既存スムーザである w-Jc および RB-GS スムーザを用いたものと対比しつつ議論する．

5.4.1 問題設定と実験環境

数値実験に用いた問題は，1 辺の長さが 1m の立方体状の空間とその中心に設置された半径 7.8cm の球体を対象とするものであり，$\boldsymbol{\rho}$ の値は球体の内部では 1，外部では 0 と設定した．最密の格子空間 $\Omega^{(h)}$ の格子数は $NX = NY = NZ = 513$ とし，1 つの V サイクルのレベル数は $L = \log_2(NX - 1) = 9$ とした．初期解ベクトルは $\boldsymbol{\phi}^{(0)} = 0$ とし，収束判定条件は m 回目の V サイクル終了後の残差 $\boldsymbol{r}^{(m)}$ と初期残差 $\boldsymbol{r}^{(0)}$ のユークリッドノルム比を用いて $\|\boldsymbol{r}^{(m)}\|_2/\|\boldsymbol{r}^{(0)}\|_2 \leq 10^{-7}$ とした．また w-Jc と RB-GS スムーザについては，プリスムージングとポストスムージングの反復回数 pr と po をともに 1 とし，mBRB-GS スムーザについては $pr = po = 2$ とした．

w-Jc と RB-GS スムーザのスレッド並列化には，格子空間を z 軸に垂直な平面でできるだけ均等に分割する 1 次元分割を用い，z 軸方向の格子点数がスレッド数に満たない下位レベルの格子空間に対しては，格子点数とスレッド数を等しくした．mBRB-GS スムーザについては，まず 3.5.4 節と同様の考え方により表 5.1 に示した L2 キャッシュの容量 512 KB を基準として，表 5.2 に示すようにレベル 6 までのブロックサイズが $nbx \times nby \times nbz = 2048$ となるように定めた．なおこの値は，1 コアで 2 スレッドが実行されて L2 キャッ

シュが 2 スレッドで共有される場合の最適値であるが，1 コアで 4 スレッドが実行される場合にも 1024 ではなく 2048 が高い性能を与える設定であることが確認されたため，スレッド数に関わらず 2048 とした．一方レベル 7 以下では，全体の格子点数が 2048 未満となるのでブロック数を 1 としており，そのためスムージングは逐次 GS 法によるものとなっている．またレベル 6 はブロック数が 2 であるため mBRB-GS スムージングを 1 スレッドで行い，レベル 4 と 5 はブロック数がそれぞれ 128 と 16 であるため最大スレッド数はその 1/2 の 64 と 8 となる．それ以上のレベルは最大 240 スレッドの並列実行が可能であり，x 方向のスレッド数は常に 1 とした上で，総スレッド数 T に応じて y 方向および z 方向のスレッド数 T_y と T_z を以下のように定めた．

$T = T_y \times T_z$		
2	1	2
4	2	2
8	2	4
16	4	4
32	4	8
60	4	15
120	8	15
240	15	16

それぞれのソルバプログラムは Fortran2003 で記述し，OpenMP 3.1 準拠のコンパイラ Intel Fortran Composer XE (version 14.0.0) を用い，最適化オプション O3, mmic, no-opt-prefetch を与えてコンパイルした．プログラムの実行には Xeon Phi 7120 を 1 個使用し，各コアにスレッドが均等に割り付けられるように環境変数 KMP_AFFINITY の値を balanced とした．

5.4.2 実験結果

表 5.3 に，各スムーザを用いたマルチグリッド法ソルバの V サイクル数と，スレッド数を 1, 60, 120, 240 としたときの計算時間（秒）を示す．また図 5.3 はこれらの結果およびスレッド数が 2, 4, 8, 16 および 32 の結果を，mBRB-Alg1 の 1 スレッド性能を基準とした台数効果の形で示したものである．

まず w-Jc スムーザについては，V サイクルあたりの計算時間の最小値（120 スレッド実行）が 0.89 秒であり，RB-GS（1.81 秒），mBRB-Alg1（1.10 秒），mBRB-Alg2（0.92 秒）のいずれよりも短い時間となっている．この結果は 5.3 節で議論したように，w-Jc のループ構造が最も SIMD 並列化に適合していることを反映している．しかし，V サイクル数が他のスムーザの 2 倍以上であるためソルバ全体での性能は低く，1 スレッドおよび 60 スレッド実行では最低の値となっている．

表 5.3: スムーザごとの V サイクル数およびスレッド数と計算時間の関係

		スレッド数			
	V サイクル数	1	60	120	240
w-Jc	18	1025.44	19.55	16.09	19.38
RB-GS	9	885.54	18.35	16.29	21.97
mBRB-Alg1	7	585.43	11.18	8.16	7.67
mBRB-Alg2	7	651.03	12.55	8.89	6.44

RB-GS スムーザに関しては，V サイクルあたりの計算時間が最長であることが悪影響を及ぼし，特に 120 スレッドおよび 240 スレッド実行の性能は最低となっている．このことは，5.3.1 節で示した SIMD 演算のスループットが w-Jc スムーザの 1/2 程度になることに加えて，3.2.2 節でも議論した低い参照局所性の問題が物理メモリのアクセスバンド幅が逼迫する多スレッド実行で顕在化しやすいことを示している．

一方 mBRB-GS スムーザを用いたソルバでは，どちらの実装についても w-Jc や RB-GS スムーザによるものよりも高い性能が，スレッド数に関わらず得られている．特に，SIMD 並列化が全く適用できない mBRG-Alg1 によるソルバの性能が既存スムーザによるものよりも優れていることは，mBRB-GS スムーザの優れた収束性とともに，良好な参照局所性が Xeon Phi での実行でも重要な要素となることを示している．

しかし，部分的にせよ SIMD 並列化が実施できている mBRB-Alg2 の性能は，全く SIMD 並列化できていない mBRB-Alg1 の性能と大きな差がなく，120 スレッド以下の実行では mBRB-Alg1 に劣る結果となっている．次節ではこの理由の詳細な解析と，その結果に基づく実装の改良について述べる．

5.4.3 mBRB-GS スムーザの性能解析と実装改良

前節で述べたように，部分的 SIMD 並列化が可能な mBRB-Alg2 の 240 スレッド実行以外での性能が，SIMD 並列化ができない mBRB-Alg1 に劣る理由を調べるために，Intel 社の性能解析ツール Vtune Amplifier を用いて mBRB-Alg2 の性能を詳細に解析した．このツールは，Xeon Phi を含む最近のマイクロプロセッサに搭載されている，プログラム実行時のさまざまなイベントの発生回数を集計する機構である Performance Monitoring Unit (PMU) から得たデータを収集して表示するためのものである．

解析の結果，120 スレッド実行では先行命令の遅延による read-after-write ハザードを意味する VPU_STALL_REG と，ロード・ストア命令のアドレス計算遅延によるストールを意味する PIPELINE_AGI_STALL の二つが，240 スレッド実行に比べて有意に多数発生していることが判明した．具体的には，

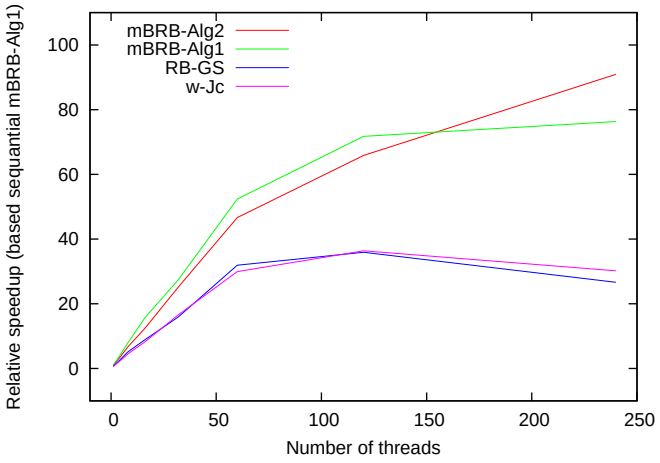


図 5.3: mBRB-Alg1 を基準とした各手法の台数効果

前者は 120 スレッド実行で 267×10^7 回であるのに対し 240 スレッド実行では 10 %程度少ない 242×10^7 回であり，後者は 120 スレッド実行では 108×10^6 回であるの対し 240 スレッド実行では検出閾値 10^6 未満であった．5.2 節で述べたように，Xeon Phi は動的な命令遅延の隠蔽を Hyper-Threading にのみ依拠しているので，コアあたり 2 スレッドの実行である 120 スレッド実行では遅延隠蔽に必要な他スレッドの命令が不足しているために，これらのハザードが顕在化しているものと考えられる．また mBRB-GS-Alg2 ではループ分割により各ループのボディが単純化されるため，コンパイラによる静的な命令スケジューリングで遅延を隠蔽することが mBRB-GS-Alg1 に比べて困難であることも一因と考えられる．

また Vtune Amplifier により，GS スムージングのループでの SIMD 命令の実行比率を調べた結果，25.9 %という低い値であることが明らかになった．この値は 6 個の最内ループの中の 1 個が SIMD 並列化されないことを勘案しても過小であるため，コンパイラが出力したアセンブリコードを確認したところ，3 次元配列のインデックスから要素のアドレスを計算するための整数演算命令がループボディに多数存在することが判明した．汎用的なマイクロプロセッサが複数の整数演算命令を浮動小数点演算命令と並列にかつアウトオブオーダで実行できるのに対し，Xeon Phi では SIMD 命令と同時に実行できる整数演算命令は 1 個であり，しかもインオーダ実行であるため，相互に依存する多数の整数演算命令の処理スループットが相対的に低い．したがってアドレス計算のための整数演算命令が前述のハザードの要因となり，mBRB-GS-Alg2 の実行速度を律している可能性が強く示唆された．

表 5.4: mBRB-Alg2-1D のスレッド数と計算時間の関係

	スレッド数			
	1	60	120	240
計算時間	562.99	10.86	7.57	5.66

以上の解析に基づき，mBRB-Alg2 の最内ループに対して，3 次元配列を仮想的に 1 次元配列とする実装改良を行った．具体的には，たとえばアルゴリズム 3 の最初のループ

```
do i = xs, xe
  tmp(i) = rho(i, j, k) + a(i, j, k) * phi(i, j, k-1)
enddo
```

を，3 次元配列 rho，a，phi を 1 次元配列で表現したものを rho1d，a1d，phi1d とし，要素 (xs,j,k) および (xs,j,k-1) に対応する 1 次元配列の要素番号を base1，base2 とした上で，以下のように変形する．

```
i1 = base1; i2 = base2
do i = xs, xe
  tmp(i) = rho1d(i1) + a1d(i1) * phi1d(i2)
  i1 = i1 + 1;  i2 = i2 + 1
enddo
```

この変形を行った mBRB-Alg2-1D では，ループボディでの配列要素のアドレス計算のための整数演算命令が最小限に抑えられ，SIMD 命令の実行比率が 25.9 %から 53.8 %に向上した．また表 5.4 と図 5.4 に示すように，スレッド数によらず mBRB-Alg1 よりも高い性能となり，240 スレッドでは mBRB-Alg1 に対して 1.36 倍，また mBRB-Alg2 に対しても 1.14 倍の速度向上となった．

5.5 まとめ

本章では，512 ビットという幅広い SIMD 演算機構を特徴とするメニーコアプロセッサ Xeon Phi を対象として，mBRB-GS スムーザを用いた高性能マルチグリッド法ソルバの実装について議論した．mBRB-GS スムーザの各ブロックに対する逐次 GS スムージングは，近似解ベクトルの更新操作がループ運搬フロー依存を伴うため SIMD 並列化が困難であるが，最内ループに対するループ分割によって得られる 6 個のループの中の 5 個にはフロー依存が生じないことを利用して，部分的な SIMD 並列化が実現できることを示した．また Xeon Phi の整数演算機構の演算スループットが一般のマイクロプロセッサよりも劣り，種々の命令遅延の隠蔽を Hyper-Threading にのみ依拠する方

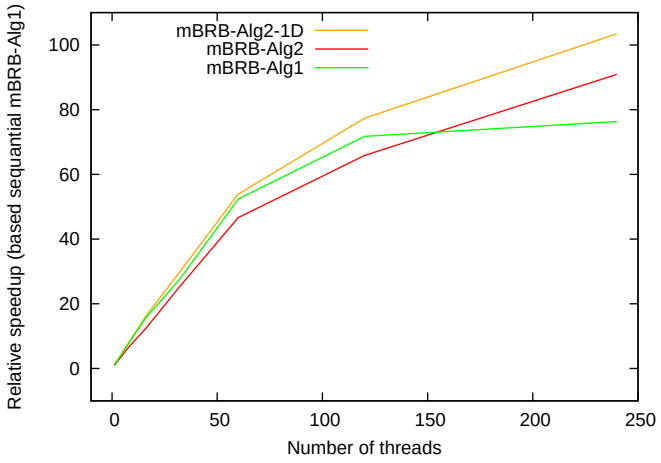


図 5.4: mBRB-Alg1 を基準とした配列の 1 次元化の効果

式であることから， 3 次元配列要素のアドレス計算のスループット不足や遅延顕在化が SIMD 並列化した mBRB-GS スムーザの性能ボトルネックであることを，詳細な実行プロファイル解析によって見出した．この解析結果に基づき，3 次元配列の 1 次元化を行ってアドレス計算を簡素化した結果，部分的な SIMD 並列化によって mBRB-GS スムーザによるソルバの性能が 1.35 倍に向上することを示すとともに，既存の SIMD 並列化可能なスムーザによるものに対しても性能面で大きな優位性があることを確認した．

第6章　まとめ

本稿では，構造格子に基づく有限差分法によってポアソン方程式を離散化して得られる連立一次方程式を，並列計算環境で高速に求解する手法の研究について述べた．この研究では，基本となる求解法は対象とする連立一次方程式に対して理想的な収束性を示すマルチグリッド法とし，その並列化に際して収束性と計算性能の両面で重要となるスムーザに着目して，2 種類の新たな並列スムーザの提案とさまざまな計算環境での実装・評価を行った．

第 3 章では，マルチグリッド法のスムーザとして収束性の面で優れていることが知られている GS 法に対して，ブロック化赤-黒順序付け法を適用した新たな並列化スムーザである BRB-GS スムーザを提案した．BRB-GS スムーザは収束性の観点では GS 法と同等であるため Hybrid スムーザよりも優れ，また RB-GS スムーザの問題点である配列要素のストライドアクセスに起因する参照局所性の低下も生じないため，これらの既存手法を用いたものよりも高性能のマルチグリッド法ソルバを構築することができる．実際，4 コアプロセッサを 4 基搭載した SMP ノードである HX600 を用いた 16 スレッド並列実行の性能は，Hybrid スムーザを用いたものの 2.00 倍，RB-GS スムーザを用いたものの 1.36 倍となり，BRB-GS スムーザの優位性が確認できた．また各ブロックに対する GS スムージングを複数回反復する改良型の BRB-GS スムーザである mBRB-GS スムーザも提案した．この方法では，ブロックサイズをキャッシュ容量に適合させることによって，複数回反復に起因する V サイクルあたりの計算時間の増加を抑えつつ V サイクル数を削減することができ，基本となる BRB-GS スムーザを用いたソルバよりもさらに高い性能を得ることができた．具体的には上記の数値実験環境での mBRB-GS スムーザを用いたソルバの性能は，Hybrid スムーザによるものの 2.88 倍，RB-GS スムーザによるものの 2.22 倍，また BRB-GS スムーザによるものの 1.64 倍となり，大きな優位性があることを確認できた．

第 4 章では，BRB-GS および mBRB-GS スムーザの分散メモリ環境への適用とその性能を，既存の Hybrid および RB-GS スムーザと対比させつつ議論した．まずこれらのスムーザを用いたマルチグリッド法ソルバの V サイクルあたりの通信量にはほとんど差がないことを，各スムーザの通信パターンを解析することで示した．したがってこれらのスムーザを用いたソルバの性能は，第 3 章の結果と同様に mBRB-GS スムーザによるソルバが最も高く，BRB-GS スムーザによるものがそれに次ぐという結果が得られた．具体的に

は，8 コアプロセッサを 2 基搭載した GreenBlade8000 のノードを 14 台用いた 216 プロセス並列実行の実験では，BRB-GS スムーザを用いたソルバの性能が Hybrid スムーザによるものの 1.56 倍，RB-GS スムーザによるものの 1.27 倍となった．また mBRB-GS スムーザを用いたソルバ性能は，Hybrid スムーザによるものの 2.60 倍，RB-GS スムーザによるものの 2.12 倍，また BRB-GS スムーザによるものの 1.67 倍となり，共有メモリ環境での結果と同様の優位性を確認できた．

第 5 章では，メニーコアプロセッサ Xeon Phi を対象とした mBRB-GS スムーザの最適化実装について議論した．Xeon Phi を用いて高い性能を得るための必須条件である 512 ビット幅の SIMD 演算を活用するために，ブロックごとの逐次 GS スムージングの最内ループに対して加算項ごとにループ分割を行うことで，一部を除いて SIMD 並列化が可能な実装手法を提案した．この方法によって，基本的かつ自然な逐次 GS スムージング実装によるものに対して 240 スレッド実行の性能が 1.19 倍向上したが，ループ分割によって 3 次元配列のアドレス計算のための整数演算命令の比率が高くなり，これが性能に悪影響を及ぼしていることも判明した．そこで，未知変数，右辺ベクトルおよび係数行列要素を格納した 3 次元配列を，GS スムージングのループでは 1 次元配列として扱い，ループ内での配列要素のアドレス計算が簡単になるような実装を行った．その結果 240 スレッド実行では，3 次元配列をそのまま用いたものに対して 1.14 倍，また基本的な実装によるものに対して 1.36 倍の，性能向上がそれぞれ得られた．また既存の w-Jc および RB-GS スムーザはともに SIMD 並列化が容易であるが，これらに対しても大きな優位性があることが，それぞれによるソルバの 3.42 倍および 3.88 倍の性能が得られたことにより確認された．

最後に，今後の研究での課題となる事項について述べる．第 4 章で述べた mBRB-GS スムーザを用いたソルバのプロセス並列化では，第 3 章で述べたスレッド並列実装では採用したスムージングと他の演算のキャッシュブロッキング実装が，プロセス間通信の大幅な煩雑化を避けるために未採用となっている．またスレッド並列化の併用，通信と計算のオーバーラップによる通信遅延の隠蔽といった，比較的一般化されている実装技法も採用していない．これらはいずれも，今後の高性能計算システムで予想されるメモリバンド幅や通信バンド幅の相対的低下やプロセッサあたりのコア数増加などへの対応技術として必要であり，優先度の高い課題として取り組む必要がある．

また第 4 章の実装での最粗格子の処理は，特定のプロセスに全プロセスの残差を収集して求解操作を行い，求まった近似解を全プロセスに分配して補間を行うというものである．第 4 章の実験環境や問題設定では，この収集・分配に要する通信量は軽微であり，また求解操作に要する時間も無視できる程度であった．しかし数万プロセスあるいはそれ以上の規模では，どちらの時間も無視できなくなる可能性がある．この問題に対して文献 [49] では，各

プロセスが担当する格子が限界まで粗くなったレベルを最粗格子とせず，プロセス数を減らすことでより粗い格子での計算ができるようにして，V サイクルのレベル数を増やす方法が提案されている．このような方法も含め，最粗あるいはそれに近い下位レベルの格子に関する並列処理法は，今後も続くと予想される並列度増加に対応するために重要な課題である．

マルチグリッド法の収束性の観点では，係数行列の性質との関係を調査することが重要な課題となっている．すなわち本稿で示した数値実験では，ポアソン方程式としては最も基本的な拡散係数が一定である問題を対象としたが，実際のシミュレーションでは空間内で拡散係数が変化する問題もしばしば用いられる．このような問題に対して，BRB-GS や mBRB-GS スムーザを用いたソルバの収束性がどのように変化するか，また既存手法に対する収束性や性能の優位性がどの程度であるかを調査することは，提案手法の問題適合性を知る上で重要である．

また mBRB-GS スムーザは BRB-GS スムーザを変形した，乗法シュワルツスムーザと考えることができる．しかし乗法シュワルツスムーザを得るという観点からは，ブロック内のスムージングを逐次 GS 法に限定する必要はなく，たとえば簡単に SIMD 並列化が可能な RB-GS 法を利用し，参照局所性に起因する RB-GS 法の欠点をブロック化と反復スムージングによって改善することも考えられる．また別のアプローチとしては，BRB などの並列順序付けを行う代わりにブロックを対象とした加法シュワルツスムーザとすることにより，mBRB-GS と同様にキャッシュブロッキングと並列化を同時に行うことも可能である．これらのスムーザにおいて，mBRB-GS スムーザと同様の収束性改善効果が得られるかを知るためには，さまざまな問題を用いた数値実験とともに変形の数理的な意味の考察が必要である．

謝辞

本研究を行う機会を与えて頂き，また直接御指導を頂きました京都大学学術情報メディアセンターの中島浩教授に厚くお礼申し上げます.

本論文をご精査頂き，貴重な御助言を頂きました，京都大学大学院情報学研究科システム科学専攻の石井信教授に深く感謝の意を表します.

本論文をご精査頂き，貴重な御助言を頂きました，京都大学大学院情報学研究科数理工学専攻の中村佳正教授に深く感謝の意を表します.

日頃の研究におきまして御指導を頂き，また本稿の執筆においても多数の的確な御助言を頂きました北海道大学情報基盤センターの岩下武史教授に厚くお礼申し上げます.

本研究を行うにあたって，貴重なご意見を頂きました京都大学学術情報メディアセンターの平石拓助教に深く感謝の意を表します.

本研究を行うにあたって，的確な御助言を頂きました京都大学学術情報メディアセンターの伊田明弘助教に深く感謝の意を表します.

最後に，本研究の遂行のみならず様々な活動に至るまでご協力頂きました，中島研究室の皆様に深く感謝いたします.

参考文献

[1] Y. Miyake and H. Usui. New electromagnetic particle simulation code for the analysis of spacecraft-plasma interactions. *Physics of Plasmas*, Vol. 16, pp. 62904–62914, 2009.

[2] T. Washio, J. Okada, S. Sugiura, and T. Hisada. Large-scale integrated model is useful for understanding heart mechanisms and developments of medical therapy. In *Ann. Intl. Conf. IEEE Engineering in Medicine & Biology Society*, pp. 2347–2350, 2009.

[3] TOP500. `http://www.top500.org/`.

[4] J. Reinders. *An overview of programming for Intel® Xeon processors and Intel® Xeon Phi coprocessors*. Intel, 2012.

[5] H. E. Kulsrud. A practical technique for the determination of the optimum relaxation factor of the successive over-relaxation method. *Communications of ACM*, Vol. 4, pp. 184–187, 1961.

[6] R. Hestenes, R. Magnus, and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, Vol. 49, pp. 409–436, 1952.

[7] G. Pöplau, R. U. Van, and K. Floettmann. 3D space charge calculations for bunches in the tracking code Astra. In *European Particle Accelerator Conf.*, Vol. 6, pp. 2203–2205, 2006.

[8] D. Takahashi. An implementation of parallel 3-D FFT with 2-D decomposition on a massively parallel cluster of multi-core processors. In *Parallel Processing and Applied Mathematics*, Vol. 6067, *LNCS*, pp. 606–614. Springer Berlin Heidelberg, 2010.

[9] J. D. Hoffman and S. Frankel. *Numerical Methods for Engineers and Scientists*. CRC press, 2001.

[10] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial Second Edition*. SIAM, Philadelphia, PA, 2000.

[11] L. Kohaupt. Basis of eigenvectors and principal vectors associated with Gauss-Seidel matrix of $\boldsymbol{A} = Tridiag[-1\ 2\ -1]$. *SIAM Review*, Vol. 40, No. 4, pp. 959–964, 1998.

[12] P. Wsseling. *An Introduction to Multigrid Methods*. John Wiley & Sons Ltd., Philadelphia, 2004. Corrected Reprint, R. T. Edwards, Inc.

[13] U. Trottenberg, C. Oosterlee, and A. Achuller. *Multigrid*. Elsevier Academic Press, 2001.

[14] W. A. Mulder. A new multigrid approach to convection problems. *Journal of Computational Physics*, Vol. 83, No. 2, pp. 303–323, 1989.

[15] A. Brandt and O. E. Livne. *Multigrid Techniques : 1984 Guide with Applications to Fluid Dynamics*, Vol. 67. SIAM, 2011.

[16] E. L. Wachspress. Extended application of alternating direction implicit iteration model problem theory. *Journal of the Society for Industrial & Applied Mathematics*, Vol. 11, No. 4, pp. 994–1016, 1963.

[17] N. Ellner and E. Wachspress. Alternating direction implicit iteration for systems with complex spectra. *SIAM Journal on Numerical Analysis*, Vol. 28, No. 3, pp. 859–870, 1991.

[18] R. E. Lynch and J. R. Rice. Convergence rates of ADI methods with smooth initial error. *Mathematics of Computation*, Vol. 22, No. 102, pp. 311–335, 1968.

[19] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, 2nd edition, 2003.

[20] A. Jennings and G. M. Malik. The solution of sparse linear equations by the conjugate gradient method. *Intl. Journal for Numerical Methods in Engineering*, Vol. 12, No. 1, pp. 141–158, 1978.

[21] R. Kettler. Analysis and comparison of relaxation schemes in robust multigrid and preconditioned conjugate gradient methods. In *Multigrid Methods*, Vol. 960, *LNM*, pp. 502–534. Springer Berlin Heidelberg, 1982.

[22] P. Sonneveld, P. Wesseling, and P. M. De Zeeuw. Multigrid and conjugate gradient methods as convergence acceleration techniques. In *Multigrid Methods for Integral and Differential Equations*, Vol. 3, pp. 117–168. Oxford UP, 1985.

[23] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation*, Vol. 31, No. 137, pp. 148–162, 1977.

[24] D. Drikakis, O. P. Iliev, and D. P. Vassileva. A nonlinear multigrid method for the three-dimensional incompressible Navier-Stokes equations. *Journal of Computational Physics*, Vol. 146, No. 1, pp. 301–321, 1998.

[25] A. Brandt, S. McCoruick, and J. Huge. Algebraic multigrid (AMG) for sparse matrix equations. *Sparsity and its Applications*, pp. 257–284, 1985.

[26] P. Bastian, G. Wittum, and W. Hackbusch. Additive and multiplicative multi-grid – a comparison. *Computing*, Vol. 60, No. 4, pp. 345–364, 1998.

[27] P. Brown, R. Falgout, and J. Jones. Semicoarsening multigrid on distributed memory machines. *SIAM Journal on Scientific Computing*, Vol. 21, No. 5, pp. 1823–1834, 2000.

[28] L. Adams and J. Ortega. A multi-color SOR method for parallel computation. In *Intl. Conf. Parallel Processing*, pp. 53–56. Citeseer, 1982.

[29] I. Yavneh. On red-black SOR smoothing in multigrid. *SIAM Journal on Scientific Computing*, Vol. 17, No. 1, pp. 180–200, 1996.

[30] T. Zeiser, G. Wellein, A. Nitsure, K. Iglberger, U. Rude, and G. Hager. Introducing a parallel cache oblivious blocking approach for the lattice Boltzmann method. *Progress in Computational Fluid Dynamics*, Vol. 8, pp. 179–188, 2008.

[31] C. C. Douglas, J. Hu, M. Kowarschik, U. Rüde, and C. Weiß. Cache optimization for structured and unstructured grid multigrid. *Electronic Transaction on Numerical Analysis*, Vol. 10, pp. 21–40, 2000.

[32] V. E. Henson and U. M. Yang. BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, Vol. 41, pp. 155–177, 2002.

[33] R. D. Falgout and U. Yang. hypre: A library of high performance preconditioners. In *Computational Science-ICCS 2002*, Vol. 2331, *LNCS*, pp. 632–641. Springer Berlin Heidelberg, 2002.

[34] M. F. Adams. A distributed memory unstructured Gauss-Seidel algorithm for multigrid smoothers. In *2001 ACM/IEEE Conf. Supercomputing*, pp. 1–14. ACM, 2001.

[35] F. Irigoin and R. Triolet. Supernode partitioning. In *15th ACM SIGPLAN-SIGACT Symp. Principles of Programming Languages*, pp. 319–329, New York, NY, USA, 1988. ACM.

[36] H. Jin, M. Frumkin, and J. Yan. The OpenMP implementation of NAS parallel benchmarks and its performance. Technical report, NAS-99-011, NASA Ames Research Center, 1999.

[37] S. Williams, D. D. Kalamkar, A. Singh, A. M. Deshpande, B. V. Straalen, M. Smelyanskiy, A. Almgren, P. Dubey, J. Shalf, and L. Oliker. Optimization of geometric multigrid for emerging multi- and manycore processors. In *Intl. Conf. High Performance Computing, Networking, Storage and Analysis*, Vol. 96, pp. 1–11, 2012.

[38] W. Gentzsch. A fully vectorizable SOR variant. *Parallel Computing*, Vol. 4, No. 3, pp. 349–353, 1987.

[39] T. Iwashita and M. Shimasaki. Block red-black ordering: a new ordering strategy for parallelization of ICCG method. *Intl. Journal of Parallel Programming*, Vol. 31, pp. 55–75, 2003.

[40] G. Behie and J. P. Forsyth. Incomplete factorization methods for fully implicit simulation of enhanced oil recovery. *SIAM Journal on Scientific and Statistical Computing*, Vol. 5, No. 3, pp. 543–561, 1984.

[41] S. Doi and T. Washio. Ordering strategies and related techniques to overcome the trade-off between parallelism and convergence in incomplete factorizations. *Parallel Computing*, Vol. 25, pp. 1995–2014, 1999.

[42] Y. Saad and M. H. Schultz. Parallel implementations of preconditioned conjugate gradient methods. In *Mathematical and Computational Methods in Seismic Exploration and Reservoir Modeling*, pp. 108–127. SIAM, 1985.

[43] P. Thoman. *Multigrid Methods on GPUs*. VDM, 2008.

[44] T. Iwashita, Y. Nakanishi, and M. Shimasaki. Comparison criteria for parallel orderings in ILU preconditioning. *SIAM Journal on Scientific Computing*, Vol. 26, pp. 1234–1260, 2005.

[45] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, Vol. 34, No. 4, pp. 581–613, 1992.

[46] C. C. Douglas, D. T. Throme, J. Hu, J. Ray, and R. S. Tuminaro. Cache aware multigrid on adaptively refined meshes. In *European Cong. Computational Methods in Applied Sciences and Engineering*, pp. 363–380, 2004.

[47] M. Frigo and S. G. Johnson. FFTW : an adaptive software architecture for the FFT. In *Intl. Conf. Acoustics, Speech and Signal Processing*, Vol. 3, pp. 1381–1384. IEEE, 1998.

[48] W. Pugh. Uniform techniques for loop optimization. In *5th Intl. Conf. Supercomputing*, pp. 341–352, New York, NY, USA, 1991. ACM.

[49] K. Nakajima. OpenMP/MPI hybrid parallel multigrid method on Fujitsu FX10 supercomputer system. In *Cluster Workshops (in Conjunction with Intl. Conf. Cluster Computing)*, pp. 199–206. IEEE, 2012.

発表論文一覧

第 3 章 - 河合直聡, 岩下武史, 中島浩. ”ブロック化赤-黒順序付け法に基づく並列マルチグリッドポアソンソルバ”, 情報処理学会論文誌, コンピューティングシステム, Vol. 5, No. 3, pp. 1–10, 2012.

- 河合直聡, 岩下武史, 中島浩. ”ブロック化赤-黒順序付け法に基づく並列マルチグリッドポアソンソルバ”, ハイパフォーマンスコンピューティングと計算科学シンポジウム論文集, 名古屋大学, pp. 107–116, 2012.

- M. Kawai, T. Iwashita, H. Nakashima and Y. Takahashi, ”Parallel Multigrid Poisson Solver based on Block Red-Black Ordering”, *Intl. Cong. Industrial and Applied Mathematics*, Vancouver, Canada, 2011.

第 4 章 - M. Kawai, T. Iwashita, H. Nakashima and O. Marques, ”Parallel Smoother Based on Block Red-Black Ordering for Multigrid Poisson Solver.”, In *Intl. Mtng. High-Performance Computing for Computational Science-VECPAR 2012*, Vol.7851, *LNCS*, Springer Berlin Heidelberg, pp. 292–299, 2013.

第 5 章 - M. Kawai, T. Iwashita and H. Nakashima, ”SIMD Implementation of a Multiplicative Schwarz Smoother for a Multigrid Poisson Solver on an Intel Xeon Phi Coprocessor.”, In *Intl. Mtng. High-Performance Computing for Computational Science-VECPAR 2014*.

- 河合直聡, 岩下武史, 中島浩. ”最新プロセッサの SIMD 演算を活用する並列化ガウスザイデルスムーザの実装方法”, ポスター発表, SACSIS2013, 仙台, 2013.